Specifiche tecniche di integrazione

CSR-D Lombardia

Revisione del Documento: **03**

Data revisione: **24/07/2026**

	Struttura	Data
Redatto da:	*Divisione - Servizi per le Entrate Regionali, Territorio, Ambiente, Mobilità – ARIA S.p.A.*	24/07/2026

Cronologia delle Revisioni

Revisione	Data	Sintesi delle Modifiche
01	03/10/2025	Prima versione del documento che descrive le API di integrazione con il CSR Digitale
02	23/01/2026	Aggiorna il documento fornendo maggiori dettagli, recepisce aggiornamenti a seguito dei test effettuati sulle API e integra dettagli sui metodi utilizzati per la tokenizzazione
03	24/07/2026	Aggiorna il documento fornendo maggiori dettagli e recepisce gli aggiornamenti effettuati sulle API

Tabella 1 - Cronologia delle revisioni

Limiti di utilizzo del documento
In base alla classificazione del documento.

Tabella 2 - Link di utilizzo del documento

Sommario

Indice delle Tabelle

Indice delle Figure

Glossario e Acronimi

Nome	Descrizione
Account	Entità dati definita da un accountId, che rappresenta ogni UserId
ABT	Account Based Ticketing
Alias o PaymentAlias	Alias Regionale, identificativo della carta di credito/debito nel sistema regionale
Anagrafica utente	Rappresenta i dati anagrafici di un utente
Anagrafica/Tessera	Identificativo dell'anagrafica della tessera nei vari sistemi (OT o CSR-D)
ARIA	Azienda Regionale per l'Innovazione e gli Acquisti S.p.A.
Autenticazione	Processo di accesso al sistema da parte di un utente con verifica delle sue credenziali
BE	Back End
CCA	Centro Controllo Aziendale, possiamo considerarlo come il BE dell'OT
CSR	Centro Servizi Regionale
CSR-D	CSR Digitale
DeviceId	Identificativo univoco del Device
DeviceToken	Ottenuto da App tramite SDK, rappresenta il device firmato con il quale si sta effettuando l'operazione di acquisto del TdV
EMV	Europay Mastercard Visa
FE	Front End
IVOL	Io Viaggio Ovunque in Lombardia
IVOP	Io Viaggio Ovunque in Provincia
OT	Operatore di Trasporto
PNR	CSR-D Codice univoco che identifica un TdV digitale integrato, rappresentato da una stringa alfanumerica generata randomicamente dal CSR-D
PSP	Payment Service Provider
SDK	Software Development Kit
STIBM	Sistema Tariffario Integrato del Bacino di Mobilità
SW	Software
TdV	Titolo di Viaggio
Tessera	Supporto digitale nominativo associato a un utente al quale sono abbinati i Titoli di Viaggio (TdV) intestati al titolare
Ticketid	Codice univoco che identifica un TdV digitale integrato post-pagato, rappresentato da una stringa alfanumerica generata randomicamente dal CSR-D
TPL	Trasporto Pubblico Locale
TSP	Token Service Provider
UserId	Identificativo univoco dell'utente nel sistema regionale

Tabella 3 - Glossario e Acronimi

1. Introduzione

1.1. Contesto di riferimento

Il **CSR Digitale**, di seguito CSR-D, costituisce il sistema centrale per la gestione della bigliettazione digitale interoperabile all'interno del contesto di mobilità regionale di Regione Lombardia. La piattaforma ha come obiettivo la realizzazione di un'architettura Account-Based Ticketing (ABT), per i processi di emissione e attivazione dei TdV, capace di garantire l'interoperabilità con i sistemi di bigliettazione già in esercizio presso gli Operatori di Trasporto attivi sul territorio lombardo, favorendo la digitalizzazione dei TdV (biglietti e abbonamenti).

Durante la prima fase di sperimentazione, il CSR-D sarà responsabile della gestione dei titoli di viaggio digitali integrati **STIBM Milano e Monza, IVOL e IVOP**, comprendendo la gestione dei titoli **pre-pagati** sull'intero perimetro regionale, ad eccezione dei TdV con tariffe agevolate, e dei titoli **post-pagati, tramite EMV**, esclusivamente all'interno dell'area STIBM Milano-Monza.

Il sistema si pone come infrastruttura condivisa e neutrale tra gli Operatori, consentendo:

- l'emissione e la gestione centralizzata dei titoli digitali integrati;
- la raccolta e gestione dei dati di incasso da vendita dei biglietti e abbonamenti digitali;
- la determinazione della miglior tariffa da applicare (Best Fare) nel caso di TdV post-pagati.

In questo contesto, il CSR-D assume il **ruolo di sistema centrale** dell'architettura complessiva, mentre i sistemi periferici sono rappresentati dalle CCA e dalle cabine di regia ITS di ciascun Operatore di Trasporto. Questo modello architetturale permette di concentrare in un unico punto le funzioni strategiche di emissione, gestione e regolazione dei titoli digitali integrati, preservando al contempo la piena autonomia operativa dei singoli Operatori all'interno dei propri domini applicativi. La vendita dei TdV sarà competenza degli OT che dovranno rivedere i loro processi per integrarsi con il CSR-D. Il CSR-D non assegna alcun numero di serie del contratto del TdV, ciascuna CCA di ogni OT può gestire il codice contratto in autonomia.

Affinché l'ecosistema possa operare in modo corretto e coerente, è indispensabile che ciascun operatore realizzi le necessarie **integrazioni tra i propri sistemi applicativi e il CSR-D**. Tali integrazioni rappresentano il presupposto fondamentale per garantire l'interoperabilità tra sistemi eterogenei, uniformare i flussi informativi e assicurare coerenza e affidabilità nello scambio dei dati. Solo attraverso un'integrazione completa sarà possibile abilitare tutte le funzionalità del CSR-D e rendere disponibili servizi avanzati a valore aggiunto, come il calcolo della miglior tariffa (best fare), apportando benefici per gli Operatori e per gli utenti finali.

1.2. Scopo del documento

Il presente documento ha l'obiettivo di fornire agli Operatori di Trasporto le **specifiche tecniche necessarie per l'integrazione con il CSR-D**, al fine di garantire una gestione uniforme, interoperabile e sicura dei titoli di viaggio digitali nell'ambito della mobilità regionale.

In particolare, il documento descrive:

- le **linee guida generali** per l'integrazione al CSR-D;
- i **processi operativi e i flussi di interazione** tra gli applicativi degli Operatori e la piattaforma centrale;
- le **specifiche tecniche di dettaglio** dei servizi esposti tramite API, con particolare riferimento ai diversi ambiti (account, biglietti, abbonamenti, post-pagato, SDK e TSP).

A supporto delle sezioni descrittive, il documento è corredato da **allegati tecnici** (*swagger*) che riportano la definizione puntuale dei web service, dei parametri richiesti e delle modalità di integrazione.

In prima istanza vengono descritte le **modalità di accesso ai gateway API**, basate sullo standard **OAuth 2.0** con flusso di tipo *Client Credentials.*

In secondo luogo, viene presentata l'**architettura generale del sistema**, che si articola in **sei moduli applicativi principali**. Ciascun modulo espone specifiche API, attraverso le quali gli Operatori possono interfacciarsi con le funzionalità del CSR-D:

- **CSR Account:** supporta la gestione degli utenti e dei relativi profili.
- **CSR Ticketing:** consente la gestione e l'emissione dei biglietti digitali pre-pagati.
- **CSR TicketingSubscription:** gestisce l'emissione e la validità degli abbonamenti digitali.
- **CSR CheckValidation:** garantisce la gestione dati di validazione e controllo.
- **CSR EMV:** gestisce i titoli digitali in modalità post-pagata con tecnologia EMV.
- **CSR SDK**: fornisce le logiche di generazione del QRcode per le applicazioni mobili degli Operatori.

Successivamente, per ciascun **ambito applicativo** verranno descritti in dettaglio i **metodi (API)** messi a disposizione dal CSR-D, suddivisi per account, servizi post-pagato e pre-pagato e SDK. Ogni sezione fornirà le specifiche tecniche relative alle singole funzionalità, incluse le modalità di invocazione, i parametri richiesti, i formati di messaggio e le regole di utilizzo.

2. Autenticazione

Il sistema di autenticazione adottato per i gateway API di Pluservice rispetta lo standard OAuth 2.0 (RFC 6749). Il flusso di autenticazione è di tipo *Client Credentials* (RFC 6749 § 4.4) e prevede il rilascio di un *access token* (RFC 6749 § 1.4) da includere in tutte le richieste verso i gateway API (*resource server*).

Per quanto riguarda OAuth 2.0, gli scenari e le modalità di utilizzo sono numerosi; in questo documento ci si limita a descrivere il flusso specifico adottato, ovvero la "*client credentials*", il cui schema è il seguente:

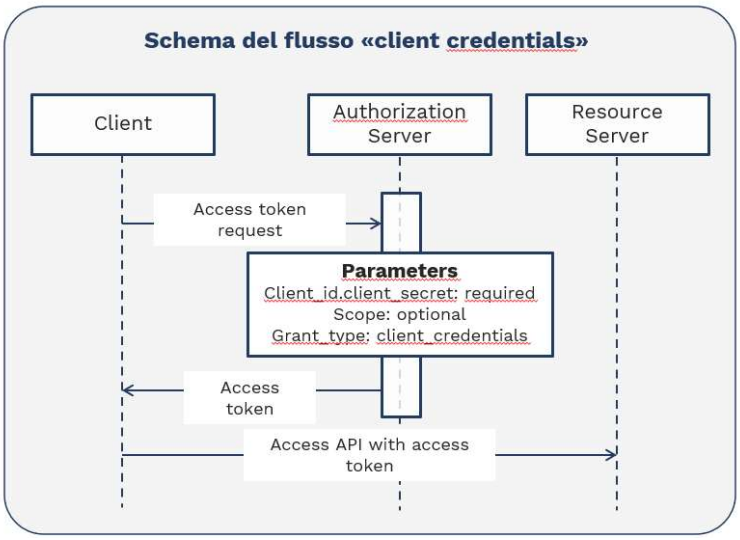


Figura 1 - Processo di autenticazione

Il Client, per accedere alle API, deve autenticarsi ed ottenere un access token. L'autenticazione del client può avvenire con diverse modalità descritte nelle specifiche dello standard "OpenID Connect Core 1.0" disponibili a questo indirizzo:

http://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication

Tra quelle che vengono descritte, il sistema supporta la modalità ***client_secret_basic*** basata su un *client_secret* fornito dall'Authorization Server. Il *client* si autentica verso l'Authorization Server in accordo alla sezione 2.3.1 di **OAuth 2.0** [RFC6749] utilizzando un "HTTP Basic authentication scheme". L'access token ottenuto ha una scadenza definita mediante il campo *expires_in* (numero intero che definisce la durata in secondi). Allo scadere della validità deve essere richiesto un nuovo token. L'access token deve essere inviato in tutte le richieste destinate alle API come Authorization http request Header di tipo Bearer.

Per ottenere un nuovo access token è necessario inviare al token endpoint una richiesta HTTP analoga alla seguente:

```
curl --location 'https://dominio.it/proxy.imomo.test/api/connect.token' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--data-urlencode 'grant_type=client_credentials' \

--data-urlencode 'client_id=OT_Client_User' \

--data-urlencode 'client_secret=xxxxxxxxxxxxxxx'
```

Si ottiene la seguente *response*:


```
{

"access_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6Ijg4QzU2MTYxMUI5NjRFMjA5REM2OTE1QjUyNDI4OUVFRjExMUU5MzBSUzI1NiIsInR5cCI6ImF0K2p3dCIsIng1dCI6ImlNVmhZUnVXVGlDZHhwRmJVa0tKN3ZFUjZUQSJ9.eyJuYmYiOjE3NTUwMDk3MjcsImV4cCI6MTc1NTAxMzMyNywiaXNzIjoiaHR0cHM6Ly9zdGFnZS5hdXRvYnVzLml0L0lkU2VydmVyLnRlc3QiLCJhdWQiOiJBUElfR2F0ZXdheSIsImNsaWVudF9pZCI6IkNTUkQuVHJlbm9yZC5Vc2VyIiwiaWF0IjoxNzU1MDA5NzI3LCJzY29wZSI6WyJBUElfR2F0ZXdheSJdfQ.YdC6OokMLFBwq75HMK7ol19vrfsGyMkLllo7kiylLorWRlGqy6UmgKYKoiPE9dlKenG81tiCbAF6yMflN4eSXdlpwI0gTBPFA41riezCmc2TiDNCEWhCho5Ni9J8t7-nbV--PsR1bbKlwGjC030Hhi0LJXfXxkHyAzKV-lPqu7oMhVLkJtzRmOxAC7FNYCMA-mNbaY893XsfpPeBetGtgoFaMcPXf-4zh6TSLWsjx4wF_PcOdXqQyx-7K3HjuOjr_hwwQqAXBYXzLAd8yWbBTGwIPhcLM1q22D58SLaWLNN-PWGnC96ZK-UFtpnd2hhS6C8XEcRiSDM-ZBYv-IB4Nw",

    "expires_in": 3600,

    "token_type": "Bearer",

    "scope": "yyyyyyyyyy"

}
```


client_id e *client_secret* sono parametri forniti dal fornitore della piattaforma CSR-D, utilizzati per autenticare gli utenti delle API (*consumer*) sul *token endpoint* e generare un *access token* valido per una specifica API in test o produzione.

client_id e *client_secret* sono confidenziali e devono essere conservati in modo sicuro da parte dell'OT, nel rispetto delle buone pratiche di sicurezza informatica e nel rispetto delle normative vigenti, prevenendo qualsiasi accesso da parte di soggetti non autorizzati. Come ulteriore misura di sicurezza, tali parametri hanno una scadenza e vengono sostituiti periodicamente o comunque quando ritenuto necessario. Per poter richiedere una nuova credenziale potrà essere aperto un ticket di assistenza tecnica. Per ogni richiesta correttamente autenticata, il *token endpoint* restituisce un oggetto JSON (RFC 6749 § 5.1) contenente un *access token* (*access_token*), il rispettivo tipo (*token_type*) e la scadenza in secondi (*expires_in*).

Disponendo di un *access token* valido, è possibile inviare richieste autenticate ad un gateway API specificando il seguente header HTTP: Authorization: *token_type access_token*

I valori *token_type* e *access_token* sono presenti nelle rispettive chiavi dell'oggetto JSON restituito dal *token endpoint* in risposta ad una richiesta autenticata con successo.

Nei casi in cui una richiesta HTTP diversa da 200, il gateway prevede i seguenti codici di errore:

HTTP Status Code	Codice di errore	Descrizione
401	**UNAUTHORIZED**	Richiesta non autorizzata
403	**FORBIDDEN**	Risorsa non disponibile

415	**UNSUPPORTED MEDIA TYPE**	Formato dell'input non supportato
413	**REQUEST ENTITY TOO LARGE**	Dimensione della chiamata superiore al limite consentito

Ogni *access token* ha una validità temporale limitata e può essere utilizzato più volte in diverse richieste verso tutte le API esposte dal gateway. **Il *token endpoint* deve essere contattato solo qualora non si disponga già di un *access token* valido.** La validità dell'*access token* è indicata nel parametro *expires_in* come documentato nelle pagine precedenti. **Sarà compito della CCA implementare le logiche per la corretta gestione del token in funzione della sua durata temporale.** A livello di integrazione tutte le chiamate prevedono in input culture e X-Correlation-ID. Quest'ultimo elemento, nonostante sia facoltativo, permette di ricevere un riferimento univoco della chiamata e diventa pertanto utile ai fini di una ricerca incrociata durante le fasi di debug. Per tale motivo si suggerisce di popolare tale campo tramite la generazione di un UUID randomico. La sessione applicativa, nel contesto della piattaforma CSR-D, rappresenta il contesto logico e temporale che lega più richieste HTTP riconducibili a uno stesso flusso di interazione utente. A differenza dell'X-Correlation-ID, che identifica univocamente ogni singola richiesta end-to-end, la sessione applicativa è solitamente gestita tramite cookie o token (es. JWT) e mantiene uno stato tra chiamate distinte, come ad esempio l'autenticazione. I due concetti sono complementari: la sessione raggruppa più operazioni, mentre l'X-Correlation-ID traccia atomicamente ogni chiamata attraverso tutti i layer architetturali.

Nello specifico, l'uso dell'header X-Correlation-ID è così declinato:

- Valorizzazione: l'header X-Correlation-ID deve essere generato dal client originante (es. app mobile, web app, sistema esterno) come un UUID randomico (es. 550e8400-e29b-41d4-a716-446655440000).
- Propagazione: tutti i componenti applicativi a valle (API gateway, microservizi, code, database) propagano lo stesso identico valore senza modificarlo, inoltrandolo nelle chiamate sincrone/asincrone verso servizi dipendenti.
- Response: ogni servizio restituisce lo stesso X-Correlation-ID nell'header della risposta HTTP, consentendo al client di correlare richiesta e risposta.

Di seguito un esempio di chiamata e risposta utilizzano l'api di gestione utenti.

Data questa chiamata per recuperare i dati di un utente esistente, in cui viene valorizzato l'header X-Correlation-ID con un UUID casuale:

```
curl --location 'http://csraccount.b2b.services.pluservice.net/api/v1/user/000238ed-bee0-4505-aeb8-fb8fbdb90f17' \

--header 'Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6Ijc0QjM4MEQ5RUMxMjY3QjFENEQ3MTg5NzE5NTZEMzA5MTc4QTEyMjlSUzI1NiIsInR5cCI6
ImF0K2p3dCIsIng1dCI6ImRMT0EyZXdTWjdIVTF4aVhHVmJUQ1JlS0VpayJ9.eyJuYmYiOjE3NjQ4NDIyNzksImV4cCI6MTc2NDg
0NTg3OSwiaXNzIjoiaHR0cHM6Ly9jbGR3ZWIuYXV0b2J1cy5pdC9JZFNlcnZlci5iMmIiLCJhdWQiOiJBUElfR2F0ZXdheSIsImN
saWVudF9pZCI6IlRyZW5vcmRfQjJCIiwiaWF0IjoxNzY0ODQyMjc5LCJzY29wZSI6WyJBUElfR2F0ZXdheSIsIlFyQ29kZVRlbGV
tYWNvX2NvbXBsZXRlX3Njb3BlIiwiU3lzRXN0ZXJub0FyaWEiXX0.WP2GTJUXeSMC79YlUH8SXgV4r8ZKs2Z8oknnXVJVa10T3dO
Rpz36MagxJz1SK45R34LggGM71nZ5vXOff-PNFqSJ_aRMq6K5KC9Lg1K1JgOasxIpeXQcK4MkkP8IpLVREimi-
BJzbGMzOgaQDEgMz5taYFIq5amMsBewALFJPXFXpA3ZKwlEcitckHVMXT3W8QVi-ONdxHmxwXMg-V93lqehttpaLOz-
N4B7GEJszaqghuZDDSFljl5zyxmeRYeEj-nMZkk30ZcnWoK4xIKJTFFh2Jr26Ox0y0D-
AO5yDCd03Ix6S8mOYX2rB8F8KTVlXwtlhC5QsiR5GpR4q6LqoEakHth-4LV3ww6-
zYZz6pRS9KzfZAvXkaoGFrqkhf4t27vLaqF94FbYoWVDKao2NT6jPkip5fEmLt5mWiokBj32ZOz7gLMIrrUflNL8NqwzJId3WnRA
UuK_whPf9bpA7IjbMxPV2GZP9vNz4ny-T7CRGqLiiofBGkCTOPLTO9s-a-
tW6DVYBX7V7FN8GrkmPnBP6FfbKWv2HHyzrB9KEwB_7t5TfTMpn_So9jrlSWAQABEducDF6goAQabcO1EMVdvPqFTQN8HhhAFDne
R0y6qegguUp4zhv6WR8zDPrMIP63pMJOhTukrcEUUQBzTraiMdU4VhTWJnALw5GwKmlro' \

--header 'X-Correlation-ID: eb97c41a-1d3c-4634-bc7e-785d1b6b2e3e' \
```

```
--header 'Culture: it-IT'
```

Si ottiene in risposta il seguente body JSON:


```
{"userId":"000238ed-bee0-4505-aeb8-
fb8fbdb90f17","prefixNumber":"+1","phoneNumber":"552115882","email":"uni.uno@unos.it","name":"Uni","
surname":"Uno","birthYear":1911,"domicileData":{"municipalityCode":"L219","province":"TO","zip":"101
40","address":"Via Carlo Augusto","houseNumber":"37/A"},"lastUpdateDateTime":"2026-05-
08T13:18:28.81"}
```


Inoltre, nella response, vengono inviati alcuni header tra cui viene restituito anche l'originario X-Correlation-ID, così che il chiamante possa correlare la risposta ottenuta alla richiesta originaria:

```
Transfer-Encoding: chunked

Content-Type: application/json; charset=utf-8

Server: Microsoft-IIS/10.0

api-supported-versions: 1

X-Correlation-ID: eb97c41a-1d3c-4634-bc7e-785d1b6b2e3e

Culture: it-IT

X-Powered-By: ASP.NET

Date: Thu, 14 May 2026 10:04:20 GMT
```

L'approccio descritto è pienamente allineato con i principi del distributed tracing e con il modello di propagazione del contesto definito dallo standard W3C Trace Context. In particolare:

- X-Correlation-ID svolge un ruolo analogo a trace-id, identificatore univoco di una traccia.
- La propagazione invariata tra servizi corrisponde al concetto di context propagation.
- l'uso di un singolo ID correlato trasversalmente ai log e alle chiamate HTTP realizza gli obiettivi di tracciabilità end-to-end

2.1. Formato data e ora

Il formato è quello previsto dallo standard https://www.w3.org/TR/xmlschema-2/#dateTime.

I servizi del CSR-D utilizzano per i parametri data/ora la notazione UTC (es. 2007-04-05T14:00Z): la data fa sempre riferimento al fuso orario ZERO. Come previsto dallo standard (ISO 8601), le date con orario possono essere indicate con questo formato: UTC: 2007-04-05T14:00Z.

2.2. Endpoint

Gli endpoint degli ambienti di pre-produzione e produzione saranno forniti agli Operatori di Trasporto in fase di avvio delle attività di integrazione, unitamente alle credenziali di accesso.

3. Architettura generale del sistema

L'architettura del sistema prevede diversi moduli applicativi e un flusso ben definito:

- L'utente accede all'app o al sito web dell'Operatore di Trasporto (OT);
- La componente di BackEnd dell'OT effettua l'autenticazione dell'utente utilizzando il flusso OAuth 2.0 verso l'Identity Server del CSR-D e ottiene un JWT (Json Web Token) di accesso;
- Il token JWT viene incluso in ogni richiesta successiva che il BackEnd dell'OT invia al Gateway del CSR-D. Quest'ultimo agisce come punto di accesso centralizzato e proxy sicuro, inoltrando le richieste all'Identity Server per la verifica della validità del token;
- In caso di esito positivo, il Gateway smista la richiesta al modulo applicativo corretto del
- Backend CSR-D (es. CSR Account, CSR Ticketing, CSR Subscription), garantendo così che l'utente possa accedere ai servizi richiesti.

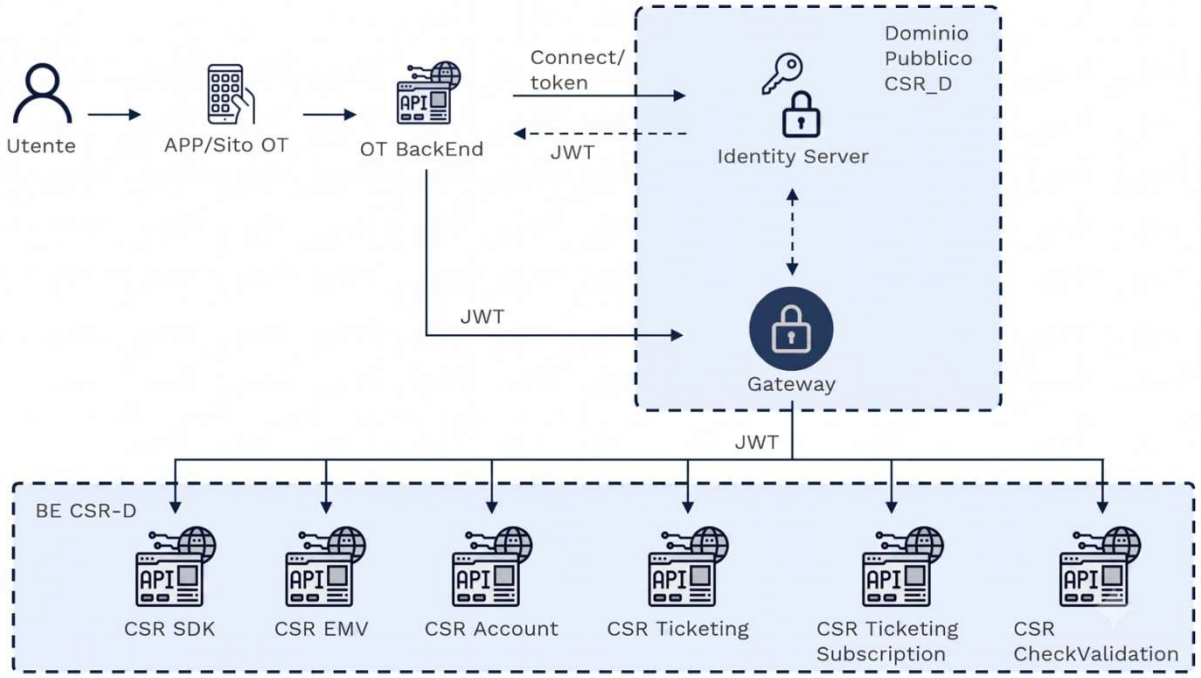


Figura 2 - Architettura generale del sistema

All'interno del CSR-D saranno presenti sei moduli, ognuno dei quali rappresenta un'area funzionale:

- **CSR Account:** Gestione degli account regionali;
- **CSR Ticketing:** Gestione dei titoli di viaggio;
- **CSR TicketingSubscription:** Gestione degli abbonamenti;
- **CSR CheckValidation:** Gestione dei dati di validazione e controllo;
- **CSR EMV:** Gestione del post-pagato;
- **CSR SDK:** Libreria software per interfacciarsi con il CSR-D.

A corredo dei moduli sopra elencati, rientra nel sistema CSR-D anche la componente TSP, in ambito EMV, utilizzata per la generazione univoca del token di pagamento (Payment Alias) a partire dai dati trasmessi dai PSP degli OT.

Di seguito viene riportata la tabella che riepiloga tutti gli allegati tecnici che descrivono i moduli del CSR-D.

Ambito	Rifermento specifiche tecniche
Account	CSRAccount Swagger 1.0.9.json
Prepagato Biglietti	CSRTicketing Swagger 1.0.9.json
Prepagato Abbonamenti	CSRTicketingSubscription Swagger 1.0.9.json
Validazione e Controllo	CSRCheckValidation Swagger 1.0.9.json
Post-pagato	CSREMV Swagger 1.0.9.json
SDK	ARIA_CSR Digitale_SDK Integration Guide_rev1.pdf
TSP	CSR TSP Swagger 1.0.2.json
Algoritmo di validazione del QRcode	ARIA_CSR Digitale_SpecificheQRcodeDinamico_rev1.pdf
Elenco codici errore (generici)	ARIA_CSR Digitale_error_code_rev2.pdf

Tabella 4 - Riferimenti alle specifiche tecniche

4. Metodi per l'integrazione (API)

Si procede descrivendo **in dettaglio i metodi (API) messi a disposizione dal CSR-D** per ciascun ambito applicativo, suddividendoli nelle seguenti categorie principali:

1. **Account:** comprendono tutte le funzionalità relative alla gestione degli utenti e dei loro profili, dall'acquisizione dei dati anagrafici alla loro eventuale modifica.
2. **Servizi pre-pagato:** coprono la gestione dei biglietti e abbonamenti digitali pre-pagati, consentendo agli Operatori di emettere, validare e storicizzare i titoli acquistati in anticipo.
3. **Servizi post-pagato:** riguardano la gestione dei titoli digitali acquistati in modalità post-pagata, mediante tecnologia EMV.
4. **Servizi per validazioni e controlli**: riguardano la gestione ed il salvataggio dei dati relativi ai controlli effettuati sui TdV ed alle validazioni ricevute dai dispositivi di campo (tornelli e validatori).
5. **SDK:** raccoglie le logiche di integrazione dedicate alle applicazioni mobili, fornendo strumenti per la generazione di QR code e la gestione delle chiavi crittografiche.

Si specifica che all'interno delle interfacce presentate nei capitoli successivi saranno presenti dei campi obbligatori e dei campi facoltativi. Saranno condivisi degli esempi di chiamate allo scopo di fornire gli opportuni chiarimenti.

4.1. Account

La prima sezione fa riferimento alla gestione degli account regionali. Vengono descritti i processi per poter registrare un account regionale, per recuperare le informazioni associate ad esso e/o per aggiornarle. Compito del CSR-D è gestire a livello centralizzato tutti gli Account che sono stati creati a partire dai dati inviati dai vari OT presenti sul territorio tramite le specifiche API; per tale motivo è compito degli OT certificare i dati dei campi chiave e verificare gli stessi (ma anche tutti i dati in input) tramite processi robusti.

All'interno del CSR-D, lo **userId** rappresenta l'identificativo univoco di un account; lo stesso codice univoco viene rilasciato a parità di **prefisso e numero di telefono**. Questo identificativo viene creato dal primo OT che ne richiederà la generazione, ma i dati dell'account potranno essere aggiornati da tutti i vari OT che utilizzeranno lo stesso userId.

Ogni OT, in fase di creazione di un account, fornirà il proprio identificativo univoco (ExternalUserGuid) associato all'account nel proprio sistema. Nel caso in cui lo userId esista già all'interno del CSR-D, perché creato da un altro OT, il CSR-D assocerà allo stesso userId più ExternalUserGuid e i relativi OT di riferimento.

Nel caso in cui l'OT abbia la necessità di associare allo stesso userId regionale un secondo ExternalUserGuid, potrà farlo richiamando il metodo di creazione con gli stessi campi chiave e passando il nuovo ExternalUserGuid. Il CSR-D salverà la doppia associazione userId, ExternalUserGuid a parità di OT. Questa situazione può verificarsi, ad esempio, qualora ci sia una richiesta di nuovo account per l'OT a parità di campo chiave.

Il modulo applicativo che consente la gestione dell'entità account regionale è CSR Account.

In questo contesto sono disponibili tre metodi (API):

- **POST /v1/User/create:** utilizzato per creare un userId.
- **GET /v1/User/{userId}:** metodo che restituisce i dati anagrafici relativi all'identificativo dell'userId;
- **PATCH /v1/User/{userId}:** utilizzato per aggiornare i dati anagrafici associati ad un determinato userId[1].

In sintesi, questo ambito applicativo permette di gestire l'intero ciclo di vita di un account regionale, dalla creazione all'aggiornamento dei dati.

POST /v1/User/Create

Il metodo User/Create permette di creare un nuovo Account Regionale. Una CCA, che non possiede nel suo sistema l'associazione tra il proprio identificativo utente e lo userId deve utilizzare questa API per richiedere la generazione di un nuovo userId o, nel caso in cui sia stato precedentemente creato da un OT per la stessa coppia ***Prefisso – Numero di cellulare****,* ottenere l'identificativo dal sistema CSR-D e salvarlo sui propri sistemi. Il CSR-D non permetterà la creazione di due account con la stessa coppia di campi chiave (Prefisso-Numero di Telefono).

Quindi, nella risposta dell'API è stato inserito un campo booleano *synchronizedData* per indicare se i dati appena inseriti sono sincronizzati con quelli presenti già a sistema.

- **synchronizedData = true**: utilizzato per una nuova creazione di un userID oppure per indicare la perfetta corrispondenza con i dati già registrati dal sistema CSR D; la verifica della corrispondenza non considera il campo ExternalUserGuid.

- **synchronizedData = false**: indica che i dati inviati non coincidono con quelli già presenti nel sistema, e, pertanto, viene inviata una richiesta all'OT al fine di confermare se i dati registrati all'interno del CSR-D sono sincronizzati. In questo caso, è a discrezione dell'OT decidere se procedere con l'aggiornamento dei dati anagrafici dell'account all'interno del CSR-D.

[1] Il campo chiave numero di telefono è composto dalla coppia prefisso + numero di telefono. Il sistema effettuerà una verifica sul suddetto campo chiave e in caso il record sia già presente restituirà un errore specifico.

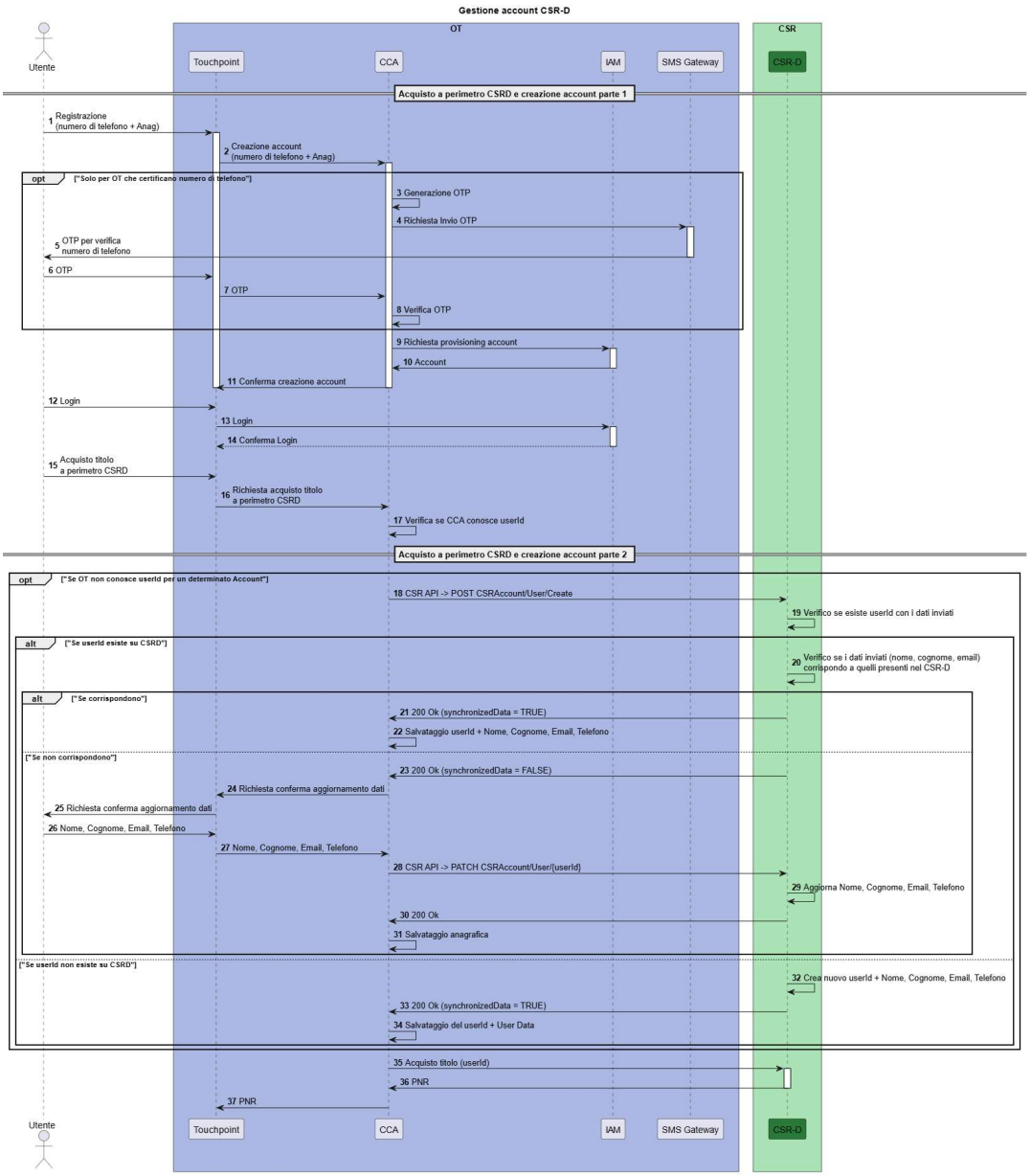


Figura 3 - Sequence diagram per creazione account

GET /v1/User/{userId}

Metodo che restituisce i dati anagrafici relativi ad uno specifico userId. Qualora, per motivi di controllo, un Operatore chiedesse informazioni circa le modifiche apportate ad un determinato userId, sarà possibile fare richiesta esplicita tramite il sistema di ticketing (messo a disposizione tramite un applicativo di back office). Il CSR-D, infatti, memorizza lo storico delle modifiche e l'OT che le ha apportate al fine di poter esportare in maniera precisa tutte le modifiche effettuate su un determinato userId.

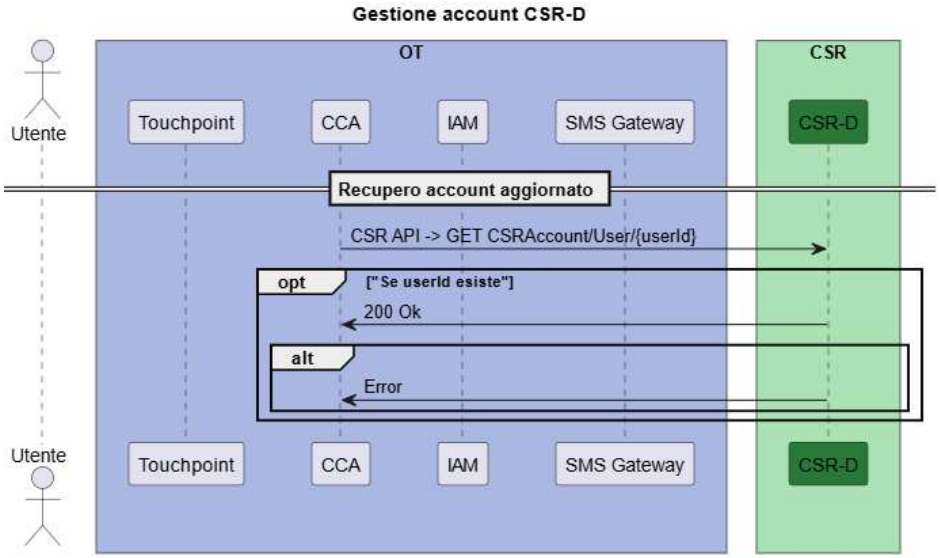


Figura 4 - Sequence diagram per ottenere i dati anagrafici di un account

PATCH /v1/User//{userId}

Il seguente metodo consente di aggiornare i dati anagrafici associati ad un certo userId. Essendo la coppia prefisso + numero di telefono campo chiave, in occasione di ogni patch, il sistema effettuerà una verifica di unicità e, qualora sia già presente un record con lo stesso campo chiave, verrà restituito un errore specifico. Nel caso, invece, in cui il campo chiave non sia già presente, il metodo aggiornerà il campo chiave e rimarrà valido lo userId precedente. Qualora si verifichi invece che, a fronte di una richiesta di aggiornamento, il campo chiave risulti già presente a sistema, verrà restituito un codice di errore specifico (KEY_FIELDS_ALREADY_EXISTING) e l'OT dovrà effettuare una richiesta di assistenza al CSR-D.

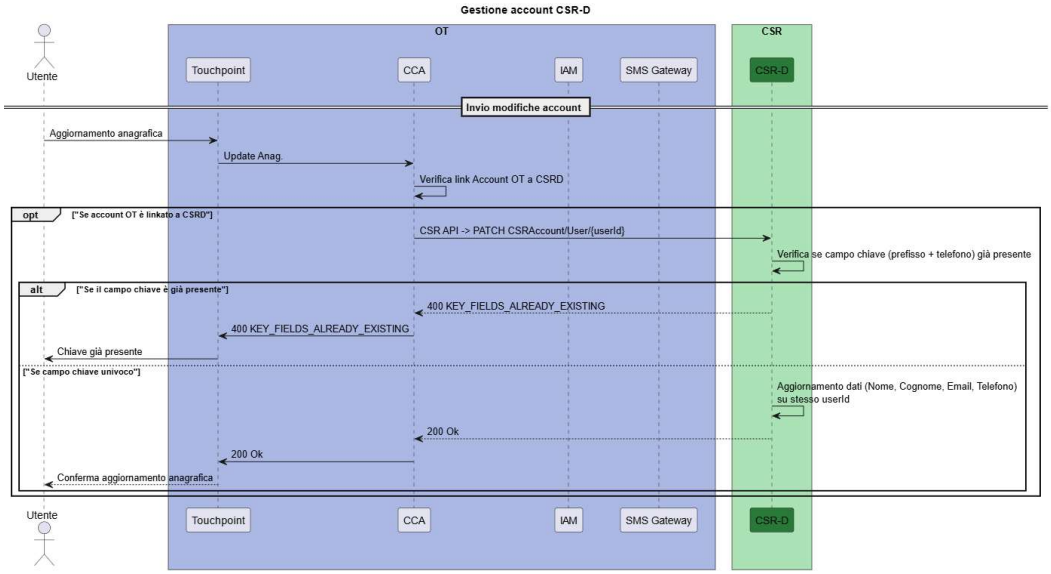
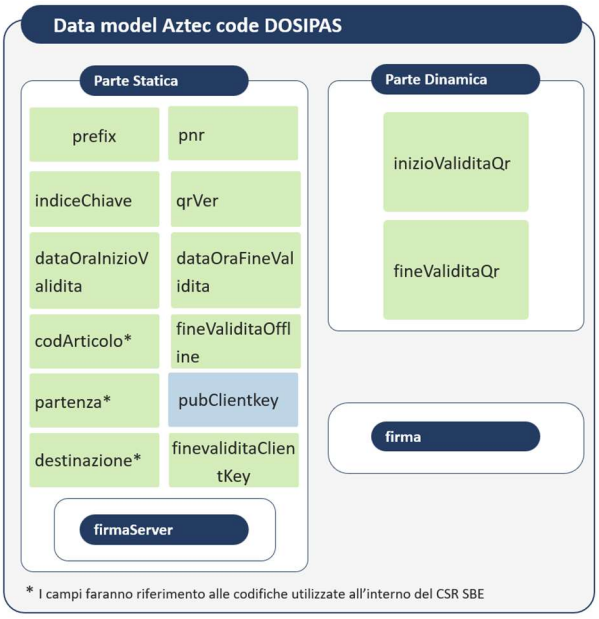


Figura 5 - Sequence diagram per aggiornare i dati anagrafici di un account

4.2. Servizi pre-pagato

I servizi pre-pagato del CSR Digitale sono progettati per gestire l'intero ciclo dei titoli digitali acquistati in modalità pre-pagata, sia in relazione agli abbonamenti che ai biglietti occasionali, oltre che le operazioni di validazione, controllo e migrazione dei TdV, comuni a entrambe le tipologie di titolo.

Il TdV sarà materializzato in forma di barcode dinamico, generato sulla base di uno specifico Data Model su standard FCB-DOSIPAS, di cui si riporta di seguito il data model con i relativi campi:



Campo	Significato
prefix	Prefisso del qrCode
pnr	Identificativo univoco del TdV
qrVer	Identificativo della versione del qrCode
indiceChiave	Identificativo della chiave di sicurezza del server
partenza*	Località di partenza del TdV nel caso di titoli venduti su origine destinazione
destinazione*	Località di destinazione del TdV nel caso di titoli venduti su origine destinazione
codArticolo*	Codice articolo del TdV digitale
dataOraInizioValidita	Inizio validità del TdV
dataOraFineValidita	Fine validità del TdV
fineValiditaOffline	Fine validità della chiave per cui è necessario tornare online per avere un nuovo qrCode
pubClientkey	Chiave pubblica del client che vuole generare il qrCode dinamico
fineValiditaClientkey	Scadenza chiave pubblica del client
firmaServer	Firma della parte statica del qrCode contenente la chiave di chi ha chiesto la generazione
inizioValiditaQr	Inizio validità del qrCode
fineValiditaQr	Fine validità del qrCode
firma	Firma complessiva del TdV per verificare autenticità del qrCode

Figura 6 - Data Model Aztec code DOSIPAS

La sezione dei servizi pre-pagato si articola in quattro aree principali:

- **Abbonamenti**: dalla configurazione tecnica per l'interazione con il CSR-D alle operazioni dedicate alla vendita del TdV.
- **Biglietti**: per la gestione completa dei biglietti digitali pre-pagati, dall'emissione allo storno dei TdV.
- **Validazione**: per gestire l'attivazione e gli eventi di validazione dei TdV.
- **Migrazione**: per gestire il cambio dispositivo dei biglietti occasionali attivi o lo stato attivo delle tessere, operazioni entrambe necessarie all'OT per garantire il corretto scarico dei TdV sulla propria mobile application.

4.2.1. Abbonamenti

Gli Operatori possono interagire con il sistema attraverso diverse API, organizzate in specifici controller dedicati a ciascuna area funzionale:

- L'**Environment Controller,** per gestire i parametri tecnici necessari all'interazione con il sistema CSR-D;
- L'**Account Controller**, per la creazione di una tessera regionale e associare/dissociare la tessera regionale all'identificativo dell'account o ad un device;
- **L'Articles Controller:** permette di ottenere la politica tariffaria per poter vendere gli abbonamenti configurati nel CSR-D;
- **Il Purchases Controller:** per eseguire tutte le operazioni dedicate alla vendita di un TdV.

Ciascun controller espone API dedicate che permettono di accedere alle funzionalità specifiche del modulo, consentendo agli Operatori di integrare in modo completo e modulare i processi pre-pagati all'interno delle loro applicazioni.

Per gli abbonamenti il metodo per ottenere il QRcode è il medesimo descritto successivamente nella sezione dedicata ai biglietti "SIGNATURES CONTROLLER". Gli abbonamenti sono attivati automaticamente in funzione delle date di validità degli stessi.

Ai fini di una miglior comprensione, i moduli (controller) e le API ad essi associate sono riportati di seguito in formato tabellare, corredati da una breve descrizione di ciascuna API.

Controller	Metodi	Descrizione
Environment controller	GET /v1/Environment/GetEnvironmentSettings	Permette di ottenere un parametro tecnico da utilizzare per le chiamate successive.
Account controller	GET /v1/Account/GetUserImage	Recupera l'immagine della tessera
	POST /v1/Account/CreateUserDataCard	Crea i dati di una tessera regionale ed effettua la verifica di unicità per i campi chiave (Codice Fiscale e prefisso – Numero Cellulare).
	POST /v1/Account/AssociateCardAccount	Il metodo si occupa di associare la tessera ad uno UserId. Prima di procedere con l'associazione vera e propria, il metodo effettuerà dei controlli di congruenza-esistenza dei dati del tesserato.
	DELETE /v1/Account/DissociateCardAccount	Il metodo si occupa di dissociare un'anagrafica/tessera dall'identificativo utente (UserId). Prima di procedere con la dissociazione vera e propria, il metodo effettuerà dei controlli di congruenza ed esistenza dei dati.
	GET /v1/Account/GetCardsListByAccount	Permette di recuperare la lista delle tessere regionali associate all'account.
	POST /v1/Account/ActivateCardDeviceAccount	Imposta come ATTIVA una tessera regionale su un deviceId, dato un determinato userId.
	PATCH /v1/Account/UpdateUserDataCardByAccount	Aggiorna le informazioni relative ad una tessera regionale.
	GET /v1/Account/GetInfoCardByAccount	Recupera le informazioni anagrafiche del tesserato, dato uno userId.
Articles controller	POST /v1/Articles/GetAllFares	Restituisce agli OT tutte le informazioni circa la vendibilità degli abbonamenti.
Purchases controller	POST /v1/Purchases/IssuePurchasesByAccount	Emette un abbonamento nel CSR-D, o STIBM o IVOL/IVOP.
	POST /v1/Articles/ConfirmPurchasesByAccount	Conferma il pagamento (l'emissione viene confermata nel CSR-D).
	GET /v1/Purchases/User/{UserId}/ Transactions/{orderId}/Status	Utilizzata in risposta alla precedente. Se completata con successo, rilascerà un PNR.

	POST /v1/Articles/RefundPurchaseByPNR	Storna l'emissione andata a buon fine rispetto ad un PNR (abbonamento).

Tabella 5 - Metodi per il pre-pagato (abbonamenti)

ENVIRONMENT CONTROLLER

GET /v1/Environment/GetEnvironmentSettings

Il primo metodo da invocare è /Environment/GetEnvironmentSettings, che permette di ottenere un parametro tecnico da utilizzare per le chiamate successive.

Il parametro ottenuto sarà poi statico e potrà essere salvato in cache, evitando così di dover ripetere la chiamata al metodo ad ogni richiesta.

Esempio

```
{
 "EnvironmentSettings":
 [
  {
   "Setting": "MOMO;ARIACLIENTEMV;0001",
   "CompanyCode": "0001",
   "CompanyName": "Ragione Sociale",
   "BusinessName ": "Ragione Sociale Commerciale",
  }
 ]
}
```

ACCOUNT CONTROLLER

L'account controller si occupa di gestire tutti i metodi relativi alla creazione, aggiornamento e recupero delle informazioni di una tessera regionale. Oltre alle operazioni di base, quali la creazione, l'aggiornamento e il recupero delle informazioni dell'account, il controller mette a disposizione dei metodi che associano/dissociano la tessera regionale all'identificativo dell'account e metodi specifici per associare una determinata tessera ad uno specifico dispositivo (device).

GET /v1/Account/GetUserImage

Il metodo GetUserImage restituisce, data in input la coppia di valori cardCode e cardNumber, la foto del tesserato.

POST /v1/Account/CreateUserDataCard

Il metodo si occupa di creare i dati di una tessera regionale garantendo l'univocità relativamente a due campi chiave, quali il "Codice Fiscale" e il "Numero di cellulare (prefisso e numero)".

Per poter creare una nuova tessera regionale l'utente dovrà aver creato precedentemente uno userId (tramite le specifiche relative al modulo Account); in seguito alla creazione, prima di poterci acquistare, l'utente dovrà associarla al proprio userId tramite il metodo "AssociateCardACcount". La logica operativa per la creazione dei dati è la seguente:

1. Verifica

Verrà effettuata una verifica di esistenza dei dati dell'anagrafica/tessera tramite i campi chiave definiti.

2. Creazione del record

Nel caso in cui non esistesse alcuna tessera per i campi chiave definiti, il metodo si occuperà di CREARE un nuovo record.

3. Recupero del record esistente

- Nel caso in cui esistesse già una tessera con i medesimi dati, il metodo si occuperà di recuperare lo stesso record impostando tutti gli oggetti synchronizedData=true.
- Se, invece, esiste una tessera con dati differenti, il metodo restituisce lo stesso record con synchronizedData=false. In questo modo l'OT potrà visualizzare le differenze e, qualora si voglia procedere ad un aggiornamento, potrà invocare il metodo PATCH (si vedano i capitoli successivi per i dettagli implementativi).

Il valore di userDataItem.userCategories andrà sempre valorizzato a null. All'interno del metodo sarà possibile allegare anche la fotografia dell'utente richiedente all'interno del campo UserPhoto; i formati supportati sono .png, .jpg e .jpeg con una dimensione non superiore a 3MB.

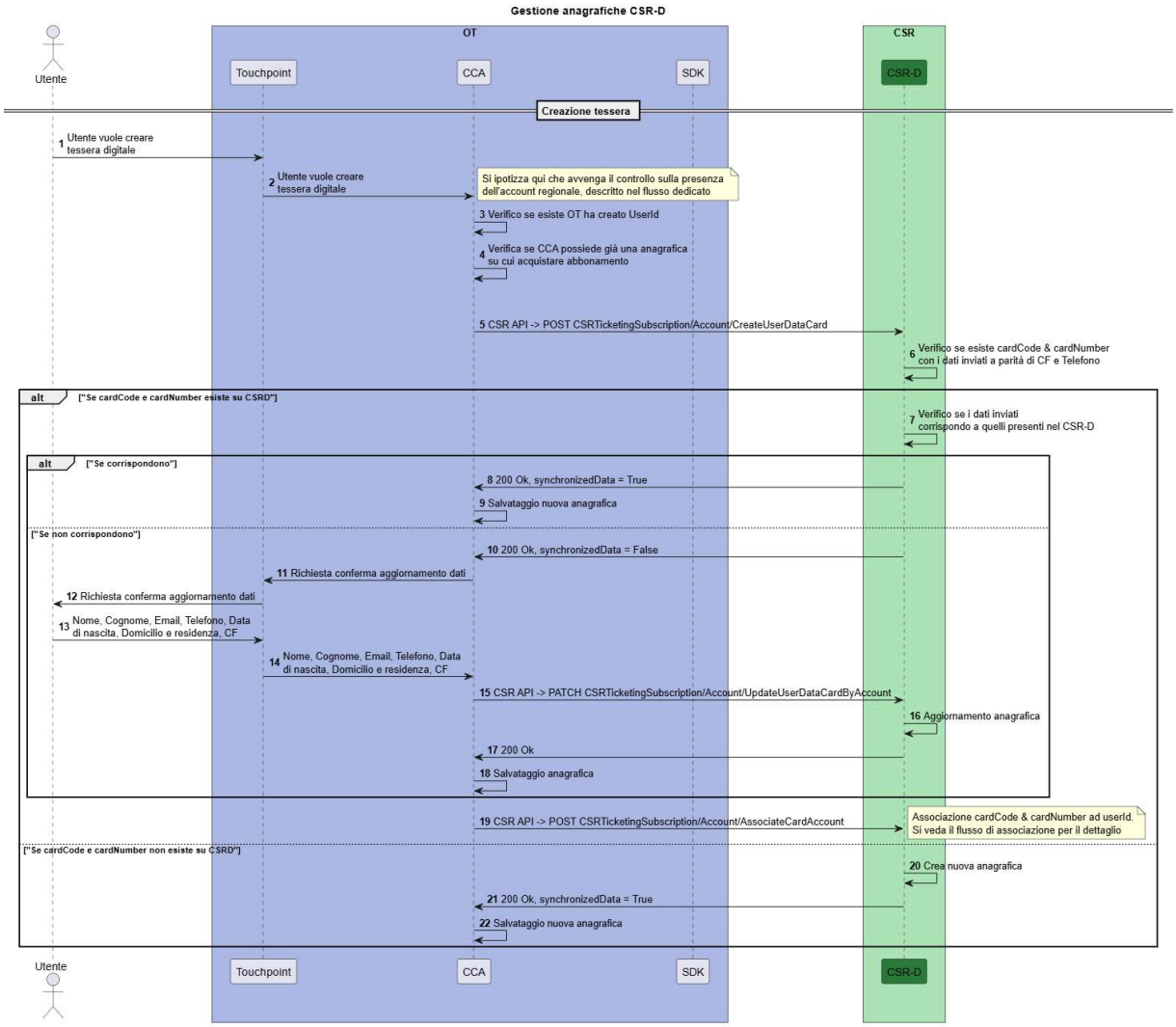


Figura 7 - Sequence diagram per flusso di creazione tessera regionale

Esempio


```
{
    "userId": "Aiu3879klda",
  "userCard": {
    "userPhotoUrl": "/Account/GetUserImage?CardCode=21&CardNumber=7003%2f8",
    "cardNumber": "7003/8",
    "cardCode": "21",
    "userDataCode": "21",
    "startValidityDate": "2025-10-01T22:00:00.000Z",
    "endValidityDate": "2075-09-30T22:00:00.000Z"
  },
  "userDataOut": {
    "synchronizedData": true,
    "name": "GGGGGGGG",
    "surname": "AAAAAAAA",
    "fiscalCode": "XXXBBB79A25H294B",
    "email": "xxx11111@lin.it",
    "phoneNumber": "2266666",
    "prefixNumber": "+39",
    "birthDate": {
      "day": "01",
      "month": "11",
      "year": 1988
    },
    "birthMunicipalityCode": "I608",
    "sex": "F"
  },
  "residenceDataOut": {
    "synchronizedData": true,
    "municipalityCode": "I608",
    "province": "AN",
    "address": "VIA ASSISI",
    "houseNumber": "7/b",
    "zip": "60019"
```

```
    },
    "domicileDataOut": {
        "synchronizedData": true,
        "municipalityCode": "H294",
        "province": "RN",
        "address": "VIA ANCONA",
        "houseNumber": "55",
        "zip": "47822"
    }
}
```


POST /v1/Account/AssociateCardAccount

Nel caso in cui l'utente sia già in possesso della tessera regionale precedentemente creata, potrà associarla al proprio account tramite tale metodo. L'associazione potrà avvenire su tutti gli Operatori di Trasporto facenti parte del CSR-D. L'unico vincolo è che l'utente finale dovrà conoscere i campi obbligatori all'associazione che vengono rilasciati in fase di creazione tessera.

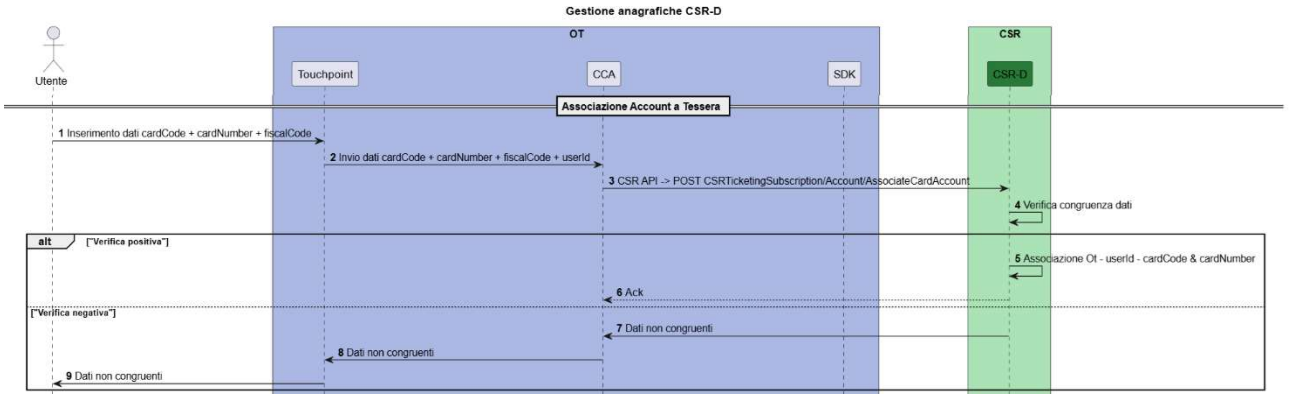


Figura 8 - Sequence diagram per flusso di associazione tessera regionale

Questo metodo dovrà, quindi, essere utilizzato prima di poter procedere verso l'acquisto di un abbonamento. I metodi dedicati all'acquisto, infatti, includono un controllo che blocca l'emissione qualora non sia preventivamente associata la tessera regionale all'account per un determinato Operatore di Trasporto. Il metodo va richiamato anche obbligatoriamente a seguito della creazione di una nuova tessera. Per questioni di sicurezza è stato previsto di inviare in ingresso al metodo i riferimenti dell'utente (userId) ed i riferimenti della tessera identificati con il campo cardCode + cardNumber ed il CodiceFiscale associato alla tessera.

DELETE /v1/Account/DissociateCardAccount

Il metodo si occupa di dissociare una tessera regionale dallo userId. Dopo tale operazione l'utente non potrà più interagire con la tessera precedentemente creata/associata, fino a che non effettui una nuova associazione. Essendo la tessera già associata ad un determinato userId, il CSR-D prevede di identificare la tessera tramite i campi cardCode&cardNumber, evitando il passaggio del codice fiscale (diversamente dal flusso di associazione).

GET /v1/Account/GetCardsListByAccount

Questo metodo permette di recuperare, dato uno userId, la lista delle tessere regionali associate all'account.

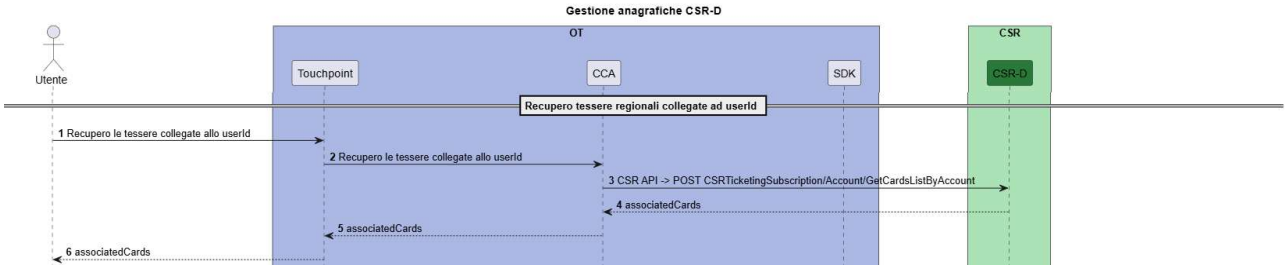


Figura 9 - Sequence diagram per flusso di recupero lista tessere per un account

Esempio


```
{
      "userId": "sda3432",
 "associatedCards": [{
      "cardNumber": "7003/8",
      "cardCode": "21",
      "ActivationStateId": true,
      "deviceId": "xxxxxxx-yyyyy-iiiii-pppppp"
   },
   {
      "cardNumber": "7003/56",
      "cardCode": "59",
      "ActivationStateId": false,
      "deviceId": null
   }
 ]
}
```


Il metodo espone un oggetto activationStateId, un enum stringa che può rappresentare tre differenti stati:

- STATO_NON_CONOSCIUTO: la tessera risulta in uno stato non conosciuto per il deviceId. Questo significa che si è verificato un qualche errore nel sistema e pertanto non sarà possibile usare quel deviceId per visualizzare il QRcode del TdV. Questo stato non compromette la visualizzazione della tessera. Al fine di fornire una risposta ai propri utenti, viene messo a disposizione degli OT un applicativo di back office;
- ATTIVO: la tessera risulta attiva per il deviceID;

- DISATTIVO: la tessera non risulta attiva per il deviceId. Questo significa che è stata dissociata e pertanto non sarà possibile usare quel deviceId per visualizzare il QRcode del TdV.

Il deviceId rappresenta il dispositivo su cui è attivata la tessera. Tale informazione serve all'OT per inserire delle possibili logiche applicative circa la possibilità o meno di visualizzare il QRcode degli abbonamenti (si ricorda che il QRcode di un abbonamento viene restituito dal CSR-D solo sul device che risulta attivo per una determinata tessera e che una tessera può essere attiva contemporaneamente solo su un device).

POST /v1/Account/ActivateCardDeviceAccount

Questo metodo, dato un determinato userId, si occupa di impostare come ATTIVA una tessera regionale e associarla ad un device. Solo una tessera con flag ATTIVA permette la visualizzazione del QRcode collegato all'abbonamento attivo; il metodo, invocabile dall'OT, risulta pertanto fondamentale nel flusso di visualizzazione dell'abbonamento. Per gli altri device sarà comunque possibile visualizzare gli abbonamenti associati ad una tessera regionale ma non sarà possibile visualizzare il QRcode. Per quanto riguarda il campo deviceToken, questo rappresenta il riferimento al device, opportunamente firmato per questioni di sicurezza, presso il quale si sta effettuando l'operazione di acquisto del TdV. Il valore deve essere recuperato tramite il metodo getSdkAuthToken specifico dell'SDK. Per i dettagli relativi all'integrazione dell'SDK si rimanda al documento di integrazione (si veda Tabella 4 - Riferimenti alle specifiche tecniche).

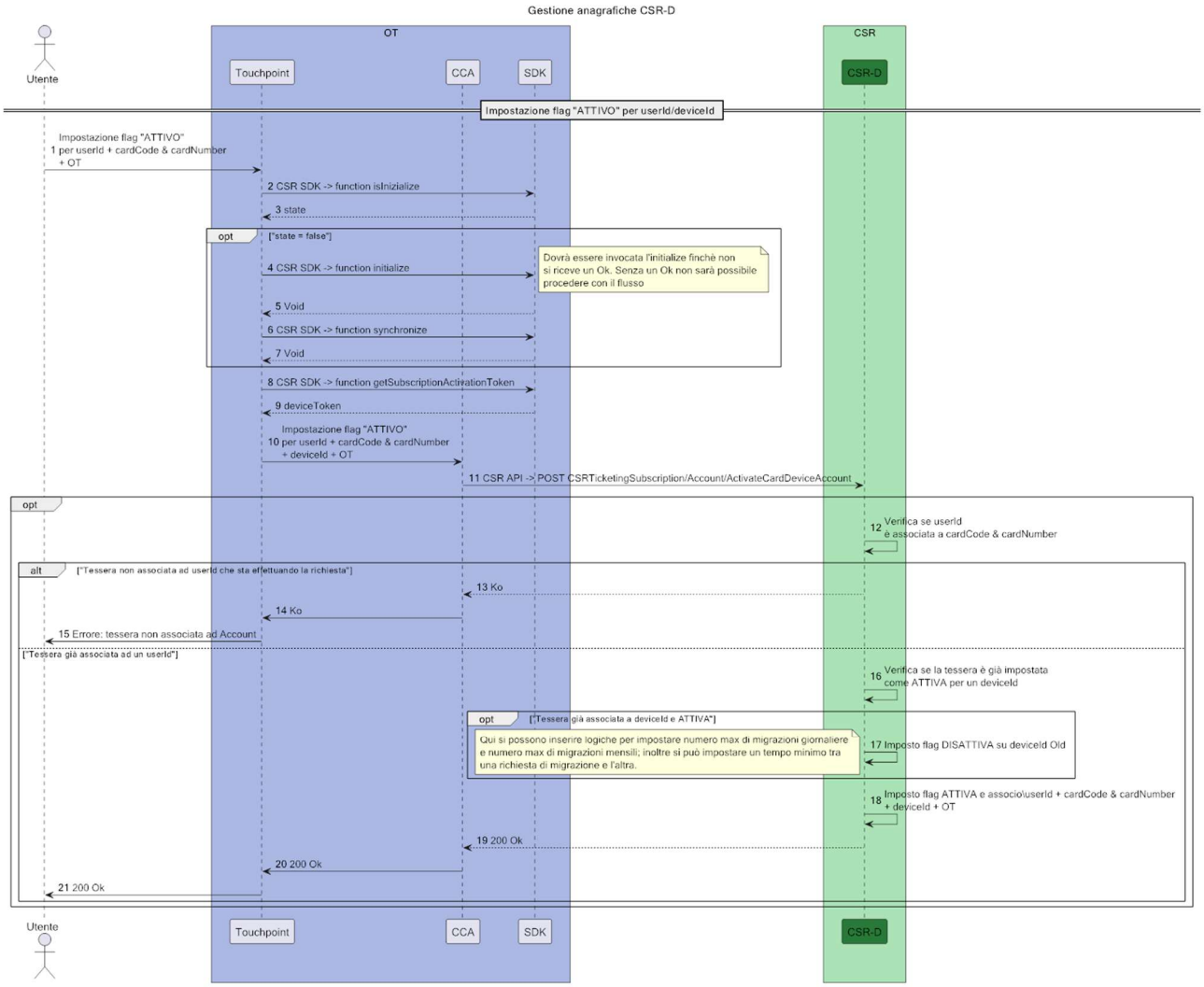


Figura 10 - Sequence diagram per flusso di attivazione tessera su deviceId

Esempio

```
{
  "userId": "0885bae2-f6a3-4864-ab33-55362cd25faf",
  "cardCode": "21",
  "cardNumber": "7003/8",
  "deviceToken": "pppppppp-xxxx-5555-ab36-poiuuyyttrre"
}
```

PATCH /v1/Account/UpdateUserDataCardByAccount

Il metodo si occupa di aggiornare le informazioni relative ad una tessera regionale.

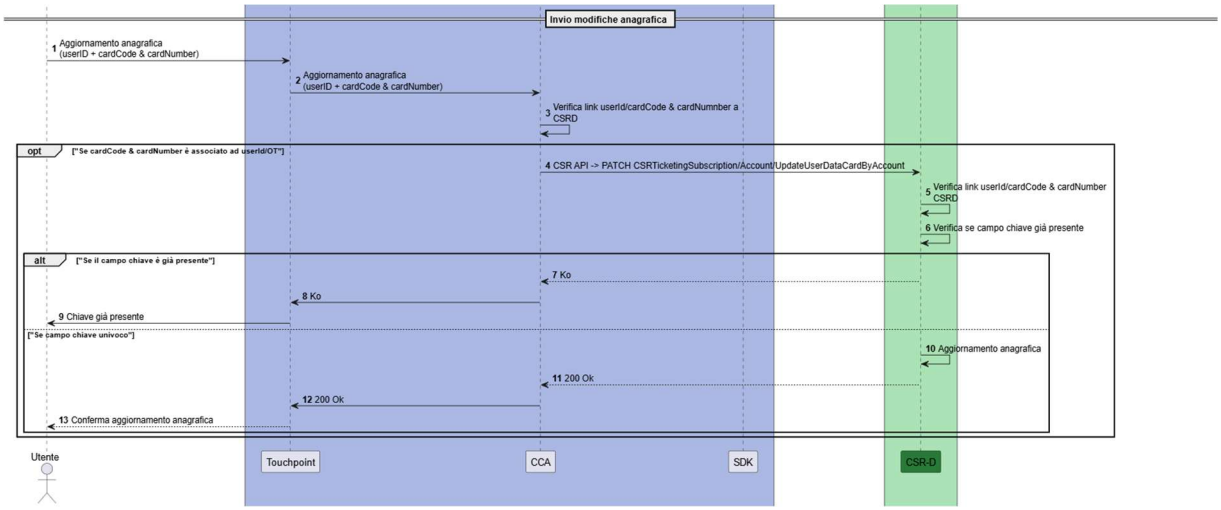


Figura 11 - Sequence diagram per flusso di aggiornamento tessera regionale

Esempio


```
{
  "userCard": {
    "userPhotoUrl": "/Account/GetUserImage?CardCode=21&CardNumber=7003%2f8",
    "cardNumber": "7003/8",
    "cardCode": "21",
    "userDataCode": "21",
    "startValidityDate": "2025-10-01T22:00:00.000Z",
    "endValidityDate": "2075-09-30T22:00:00.000Z"
  },
  "userDataOut": {
    "name": "GGGGGGGG",
    "surname": "AAAAAAAA",
    "fiscalCode": "XXXBBB79A25H294B",
    "email": "xxx11111@lin.it",
    "phoneNumber": "2266666",
    "prefixNumber": "+39",
    "birthDate": {
      "day": "01",
      "month": "02",
      "year": 1970
    },
    "birthMunicipalityCode": "I608",
    "sex": "F"
```

```
    },
    "residenceDataOut": {
        "municipalityCode": "I608",
        "province": "AN",
        "address": "VIA CONTADINO",
        "houseNumber": "55/9",
        "zip": "60019"
    },
    "domicileDataOut": {
        "municipalityCode": "H294",
        "province": "RN",
        "address": "VIA FAVOLETTA",
        "houseNumber": "36",
        "zip": "47822"
    }
}
```


GET /v1/Account/GetInfoCardByAccount

Questo metodo consente di recuperare le informazioni anagrafiche del tesserato a partire da una tessera associata ad un determinato userId. Per garantire un controllo formale, l'input della richiesta deve includere userId, cardCode e cardNumber così che il sistema possa verificare che la tessera sia effettivamente associata all'account indicato, aumentando l'affidabilità del processo di recupero delle informazioni.

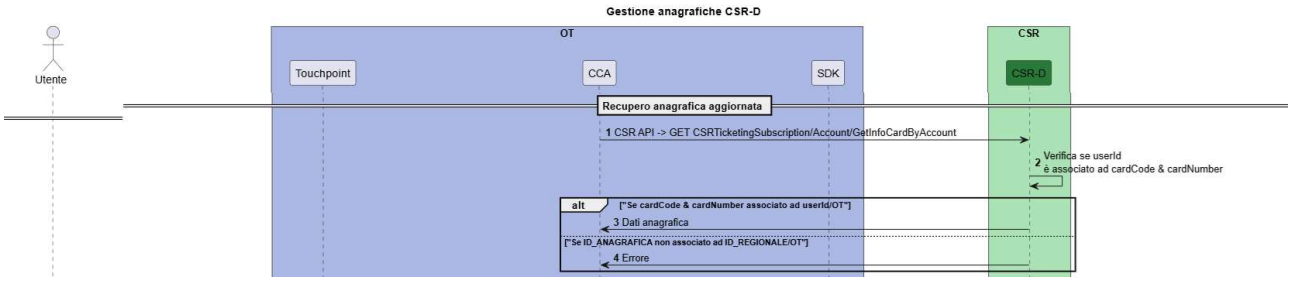


Figura 12 - Sequence diagramm per flusso di visualizzazione dati di dettagli di una tessera regionale

Esempio


```
{
    "userCard": {
        "userPhotoUrl": "/Account/GetUserImage?CardCode=21&CardNumber=7003%2f8",
        "cardNumber": "7003/8",
        "cardCode": "21",
```

```
        "userDataCode": "21",
        "startValidityDate": "2025-10-01T22:00:00.000Z",
        "endValidityDate": "2075-09-30T22:00:00.000Z"
    },
    "userDataOut": {
        "name": "GGGGGGGG",
        "surname": "AAAAAAAA",
        "fiscalCode": "XXXBBB79A25H294B",
        "email": "xxx11111@lin.it",
        "phoneNumber": "2266666",
        "prefixNumber": "+39",
        "birthDate": {
            "day": "01",
            "month": "02",
            "year": 1970
        },
        "birthMunicipalityCode": "I608",
        "sex": "F"
    },
    "residenceDataOut": {
        "municipalityCode": "I608",
        "province": "AN",
        "address": "VIA CONTADINO",
        "houseNumber": "55/9",
        "zip": "60019"
    },
    "domicilDataOut": {
        "municipalityCode": "H294",
        "province": "RN",
        "address": "VIA FAVOLETTA",
        "houseNumber": "36",
        "zip": "47822"
    }
}
```

ARTICLES CONTROLLER

POST /v1/Articles/GetAllFares

Questo metodo restituisce in output un file zip che contiene un file di testo con tutte le informazioni utili per la vendita di un abbonamento. Il metodo permette quindi a un OT di scaricare tutta la politica tariffaria e di salvarla sui propri sistemi al fine di poter tariffare gli abbonamenti vendibili in maniera totalmente autonoma, senza quindi invocare il CSR-D. Il file con tutti i dettagli è in formato TXT contenente struttura JSON, inserito all'interno di uno ZIP rinominato in questo formato:

AllFares _ @partita _ yyyymmdd _ hhmmss . zip

Il parametro @partita è un numero intero che identifica la partita di generazione all'interno del CSR-D. La partita identifica la versione del tariffario; pertanto, richiedendo in momenti diversi un file aggiornato è possibile individuare con semplicità la necessità di recepire modifiche sui sistemi degli OT. Dato che l'applicazione delle modifiche dipende dalle validità tariffarie, si suggerisce di acquisire in anticipo le tariffe aggiornate effettuando la chiamata di aggiornamento quando il CSR -D/agenzia comunicherà la disponibilità delle tariffe aggiornate.

Il nome di un file di esempio è *AllFares_123_20250822_104027*.zip. Il file TXT contenente struttura JSON segue lo stesso approccio.

Nel caso in cui un titolo di viaggio dovesse essere rimosso dalla vendita, questo non sarà più restituito all'interno del nuovo file generato. Nel caso l'utente disponesse in app di TdV non più validi, sarà onere dell'OT rimuovere tali TdV dalla disponibilità dell'utente.

Si riportano di seguito le specifiche del file TXT contenente struttura JSON con le informazioni per poter vendere un abbonamento. Si consiglia di effettuare l'aggiornamento settimanalmente.

Nome parametro	Tipo	Obbligatorio	Note
GroupingArticles	List<GroupingArticle>	yes	Lista dei raggruppamenti articoli
ArticlesData	List<ArticleData>	yes	Lista di oggetti di tipo ArticleData in cui sono contenuti tutti i campi che identificano l'entità abbonamento (es. shortdescription, ...)

<Fare>

Nome parametro	Tipo	Obbligatorio	Note
Code	String	yes	Codice tariffario
ValidityId	String	yes	Codice della validità tariffaria, associato alle date di validità estratte ed associate al codice tariffario in esame
Description	String	no	Descrizione relativa al codice tariffario
StarDate	Datetime	yes	Data inizio validità tariffaria
EndDate	Datetime	yes	Data fine validità tariffaria

<GroupingArticle>

Nome parametro	Tipo	Obbligatorio	Note
Code	String	no	Codice raggruppamento articoli (es. MEN, SET, ANN, TRI)

Description	String	no	Descrizione del raggruppamento articoli.
WebDescription	String	no	Descrizione generica pubblica associata al codice raggruppamento articoli.
WebAddingDescription	String	no	Descrizione da utilizzare come per ogni raggruppamento articoli selezionato.

<ArticleData>

Nome parametro	Tipo	Obbligatorio	Note
Code	String	yes	Identificativo articolo
Description	String	no	Descrizione articolo
ShortDescription	String	no	Descrizione breve dell'articolo (max 25 caratteri)
WebDescription	String	no	Descrizione dell'articolo
WebDescriptionExt	String	no	Descrizione estesa dell'articolo
Note	String	no	Note articolo
CodGroupingArticle	String	no	Codice raggruppamento articoli
ServiceTypeId	Integer	no*	0 = abbonamento su percorso (STIBM); <>0 abbonamento a prezzo fisso (IVOL e IVOP); Se=0 dovranno essere valorizzati anche i parametri "DepartureCode", "ArrivalCode", "SolutionId e NrNodes"
FareCode	String	Yes	Codice tariffario a cui è associato il codice articolo
DepartureCode	String	no*	Codice nodo di partenza: valorizzato solo per abbonamenti su percorso (ServicetypeId=0). Utilizzato per tutti i titoli STIBM che prevedono due zone di viaggio (es- Mi1-Mi3)
ArrivalCode	String	no*	Codice nodo di arrivo: valorizzato solo per abbonamenti su percorso (ServicetypeId=0). Utilizzato per tutti i titoli STIBM che prevedono due zone di viaggio (es- Mi1-Mi3)
SolutionId	String	no*	Identificativo soluzione: è una stringa composta e valorizzata solo se ServiceTypeId=0
NrNodes	Integer	no*	Numero di cambi. I valori ammessi sono 0, 1, 2, 3: **obbligatorio** se l'abbonamento è su percorso (STIBM) cioè con ServiceTypeId=0

ValidityArea	Obj<ValidityArea>	no	Area di validità dell'articolo: indica se l'articolo è valido per comune, provincia o regione. Questo parametro può essere valorizzato solo nel caso di articolo abbonamento non su percorso (IVOL e IVOP), cioè se ServiceTypeId>=0
Profiles	List<Profile>	yes	Lista delle categorie utente associate all'articolo. In base alla configurazione del CSR-D sarà definita di default la categoria Ordinaria.
Validities	List<Validity>	yes	Lista di intervalli di validità (data inizio e fine), valorizzate solo se IsRolling è false
IsRolling	Boolean	no	Se=true, l'articolo in esame è di tipo rolling: per essere venduto si possono ottenere le informazioni dall'oggetto validityRolling.
ValidityRolling	List<ValidityRolling>	yes	Lista contenente le specifiche di validità per gli abbonamenti rolling espresse in giorni, mesi o anni.

no*: nel caso di abbonamento su percorso dovranno essere valorizzati i seguenti parametri: "ServiceTypeId"=0, "DepartureCode", "ArrivalCode", "SolutionId e NrNodes"

<ValidityArea>

Nome parametro	Tipo	Obbligatorio	Note
MunicipalityName	String	no	Nome comune. Utilizzato per la gestione dei titoli IVOL e IVOP.
DistrictName	String	no	Nome provincia. Utilizzato per la gestione dei titoli IVOL e IVOP.
RegionName	String	no	Nome regione. Utilizzato per la gestione dei titoli IVOL e IVOP.

<Profile> ****

Nome parametro	Tipo	Obbligatorio	Note
Code	String	yes	Codice categoria utente
Description	String	no	Nome categoria utente
Title	String	no	Titolo categoria
WebDescription	String	no	Descrizione categoria web
WebDescriptionWebExt	String	no	Descrizione categoria web estesa

** Il CSR-D in questa fase gestirà solamente i profili ordinari

<Validity>

Nome parametro	Tipo	Obbligatorio	Note
StarDate	Datetime	no	Data inizio validità abbonamento (ISO8601)
EndDate	Datetime	no	Data fine validità abbonamento (ISO8601)
Price	Decimal	no	Importo abbonamento relativamente al range di validità a cui fa riferimento

<ValidityRolling>

Nome parametro	Tipo	Obbligatorio	Note
Day	Integer	no	Validità abbonamento isRolling espresso in Giorni
Month	Integer	no	Validità abbonamento isRolling espresso in Mesi
Year	Integer	no	Validità abbonamento isRolling espresso in Anni
StarDate	Datetime	no	Data inizio validità abbonamento rolling (ISO8601)
EndDate	Datetime	no	Data fine validità abbonamento (ISO8601)
Price	Decimal	no	Importo abbonamento rolling

Tabella 6 - JSON GetAllFares contenente la politica tariffaria CSR-D

PURCHASE CONTROLLER

Il puchases controller si occupa di eseguire tutte le operazioni dedicate alla vendita di un TdV.

Nello specifico il flusso prevede:

- Invocazione del metodo issuePurchasesByAccount da parte degli OT per poter emettere uno o più abbonamenti e ricevere un orderId, ossia l'identificativo univoco che identifica un ordine di acquisto al CSR-D;
- L'OT ha il compito di effettuare il pagamento verso i propri sistemi di acquiring;
- Qualora il pagamento si sia concluso con successo, l'OT dovrà invocare il metodo confirmPurchasesByAccount. Tale metodo permette di confermare sul CSR-D un determinato orderId. Il metodo in caso di esito positivo restituisce un HTTP status code 202 con il relativo URL di redirect per ottenere lo stato della transazione.

Gli abbonamenti venduti dal CSR-D non necessitano di attivazione e pertanto, in funzione delle date di validità definite durante il processo di acquisto, risulteranno automaticamente attivi nel periodo relativo alle date di validità. L'Operatore di Trasporto, in seguito alla conferma della vendita, potrà gestire in autonomia i processi di scarico attraverso i propri canali digitali. Per la visualizzazione del QRcode collegato ad un abbonamento acquistato è possibile utilizzare l'SDK. Tutti i dettagli sono riportati nel metodo ActivateCardDeviceAccount.

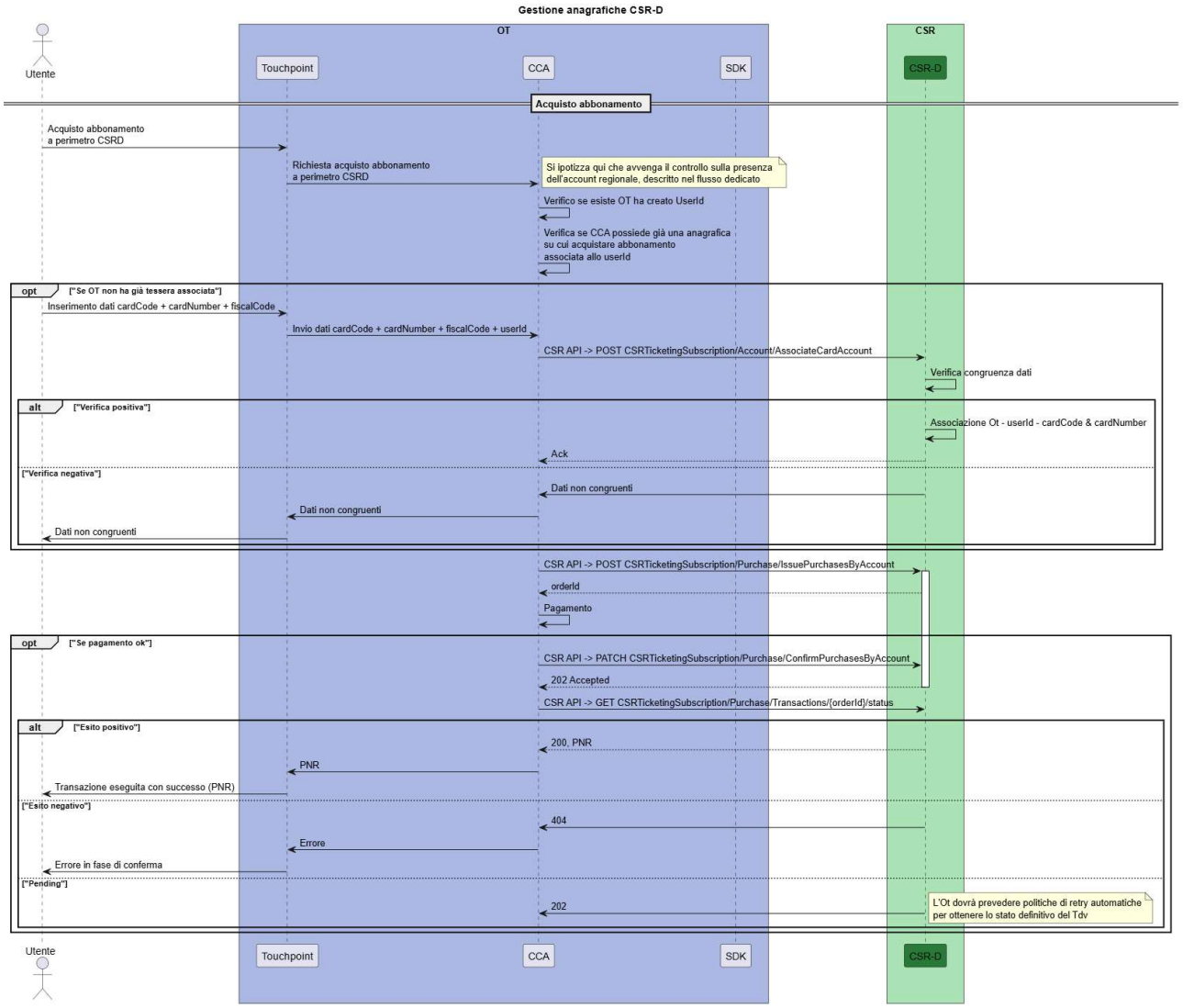


Figura 13 - Sequence diagramm per flusso di acquisto abbonamento

POST /v1/Purchases/IssuePurchasesByAccount

Il metodo si pone l'obiettivo di emettere un abbonamento nel CSR-D. Per poter invocare correttamente il metodo dovrà essere effettuata prima la chiamata al metodo GetAllFares, che si occupa di dare agli OT tutti i dati utili al fine di invocare il metodo di emissione. Fondamentalmente si possono avere due scenari:

- Tutti gli abbonamenti in area STIBM sono venduti con ServiceTypeId=0 e relativi departureCode/arrivalCode;
- Tutti gli abbonamenti IVOL e IVOP sono venduti con ServiceTypeId dedicati.

Esempio


```
{
 "orderId": 20000027,
 "totalPrice": 461.25,
 "purchaseIssueResultItems": [
  {
   "purchaseId": 20001149,
```

```
    "purchaseCode": "9081/15",
    "pnr": "AA1234567",
    "startValidityDate": "2024-05-05T22:00:00.000Z",
    "endValidityDate": "2025-05-04T22:00:00.000Z",
    "validityDesc": "Dal 06/05/2024 al 05/05/2025",
    "routeDesc": "Prezzo fisso INTERA RETE",
    "cardCode": "20000019",
    "cardNumber": "20/8",
    "carrierCode": "0109",
    "serviceTypeId": 1,
    "departureCode": "mi1",
    "arrivalCode": "mi3",
    "articleCode": "F12",
    "articleDesc": "Abb. Famiglia 360 gg Intera rete",
    "extendedArticleDescription": "Abb. Annuale Famiglia Intera rete",
    "price": 450,
    "userPrice": 450,
    "startCardValidityDate": "2024-04-29T22:00:00.000Z",
    "endCardValidityDate": "2029-04-28T22:00:00.000Z"
   }
 ]
}
```


PATCH /v1/Purchases/ConfirmPurchasesByAccount

Dopo il completamento del pagamento, l'OT deve invocare il metodo di conferma /v1/Purchases/ConfirmPurchasesByAccount che, a partire dall'orderId e dall'userId, ha il compito di validare l'emissione dei titoli acquistati. Quindi, questo metodo si occupa di confermare tutti i **purchaseId** collegati all'ordine: l'operazione viene eseguita in modo transazionale, quindi deve andare a buon fine per tutti i titoli. Se anche uno solo dei purchaseId (nel caso di un acquisto multiplo possono infatti essere più di uno) non dovesse essere confermato, l'intera operazione fallisce e il metodo restituisce un errore. Qualora l'utente abbia già effettuato il pagamento sarà possibile effettuare dei retry ed eventualmente stornare il pagamento. Il processo e il numero di retry saranno definiti da ciascun operatore. Verrà messo a disposizione un applicativo di BO per la richiesta di dettagli in caso di errore.

Gli ordini non confermati non saranno considerati dal CSR-D nelle rendicontazioni e nel clearing.

Esempio


```
{
  "userid": "xxxxxyyyyyyyyyyx",
```

```
  "orderId": 20000027,
  "purchasesCardsList": [
   {
    "purchaseId": 20001149,
    "cardCode": 21
   },
   {
    "purchaseId": 20001150,
    "cardCode": 22
   }
  ]
}
```


GET /v1/Purchases/User/{UserId}/Transactions/{orderId}/Status

L'API di status deve essere invocata in funzione della risposta 202 della confirmPurchasesByAccount. Solo in funzione di una chiamata completata con successo verrà rilasciato un PNR, ovvero l'identificativo univoco collegato ad un TdV.

HTTP Status Code	Messaggio	Descrizione
202	Pending	Significa che il CSR-D sta ancora confermando l'emissione del TdV
404	Richiesta non confermata	Significa che il CSR-D non è riuscito a confermare l'emissione del TdV
200	Richiesta confermata	Vengono restituiti tutti i dettagli del TdV

Tabella 7 - HTTP Status Code Get Status

Esempio


```
{
  "orderId": 2,
  "totalPrice": 460.0000,
  "purchasesCardsOut": [
    {
      "cardNumber": "7004/13",
      "cardCode": "16",
      "userDataCode": "16",
      "carrierCode": "1381",
      "name": "NOME",
      "surname": "COGNOME",
```

```
        "fiscalCode": "AAABBB79A25H294L",
        "startValidityDate": "2025-09-11T22:00:00.000Z",
        "endValidityDate": "2075-09-10T22:00:00.000Z",
        "confirmPurchasesOut": [
          {
            "purchaseId": 52,
            "purchaseCode": "7003/6",
            "pnr": "AP06E74F7",
            "carrierCode": "1372",
            "startValidityDate": "2025-09-30T22:00:00.000Z",
            "endValidityDate": "2026-09-29T22:00:00.000Z",
            "serviceTypeId": 0,
            "departureCode": "MI6",
            "departureDesc": "MI6",
            "arrivalCode": "MI8",
            "arrivalDesc": "MI8",
            "validityDesc": "Dal 01/10/2025 al 30/09/2026",
            "routeDesc": null,
            "articleCode": "8822",
            "articleDesc": "Annuale Mi6-Mi8",
            "extendedArticleDescription": "Annuale (mesi solari) - Abbonamento annuale person",
            "articleDescWeb": null,
            "extendedArticleDescriptionWeb": null,
            "webNotes": null,
            "price": 460.0000,
            "userPrice": 460.0000
          }
        ]
      }
  ]
}
```

POST /v1/Purchases/RefundPurchaseByPNR

Il metodo si occupa di stornare l'emissione andata a buon fine rispetto ad un PNR (abbonamento). Nel caso in cui l'utente abbia acquistato più abbonamenti nello stesso orderId, il metodo di storno va invocato per ciascun PNR emesso. Lo storno sarà abilitato solo dallo stesso OT che ha effettuato la vendita.

Il CSR-D non gestirà la parte fiscale che rimane in carico agli Operatori di Trasporto, dal momento che saranno questi ultimi a vendere il titolo di viaggio digitale. Gli OT potranno visionare tramite appositi report la lista dei titoli rimborsati e che non sono pertanto considerati nelle logiche di clearing.

Esempio


```
{
  "pnr": "BB2215453",
  "purchaseId": 2356554,
  "purchaseCode": "7005/45778"
}
```

4.2.2. Biglietti

Gli Operatori possono interagire con il sistema attraverso diverse API, organizzate in specifici controller dedicati a ciascuna area funzionale:

- **Catalog Controller**: permette ad un operatore di ottenere la lista di tutti i biglietti acquistabili tramite il CSR-D;
- **Confirmation Controller**: per la conferma dei titoli emessi;
- **Issue Controller**: per l'emissione di un TdV occasionale acquistato con autenticazione o in modalità guest, sia tramite pagina web/app OT sia da retail;
- **Redemption Controller**: per validare un PNR sia per TdV occasionale acquistato con auteticazione che guest, associa il PNR a un userID/deviceID;
- **Refund Controller**: per stornare l'emissione andata a buon fine di un PNR;
- **Validation controller**: per l'attivazione di un TdV e, in caso di mobile validation, per l'invio delle autovalidazioni e attivazioni;
- **Migrations controller**: per spostare un titolo attivo su un nuovo device;
- **Signatures controller**: per le chiavi pubbliche e la verifica della firma del QRcode.

Ciascun controller espone API dedicate che permettono di accedere alle funzionalità specifiche del modulo, consentendo agli Operatori di integrare in modo completo e modulare i processi pre-pagati all'interno delle loro applicazioni.

Ai fini di una miglior comprensione, i moduli (controller) e le API ad essi associate sono riportati di seguito in formato tabellare, corredati da una breve descrizione di ciascuna API.

Moduli	Metodi	Descrizione
Catalog controller	GET /v1/catalog	Permette ad un OT di ottenere tutti i biglietti acquistabili dal CSR-D.
Confirmation controller	PATCH /v1/orders/{orderId}/confirm	Conferma un determinato orderId.
	GET /v1/orders/{orderId}/status	Identifica lo stato di un orderId.
Issue controller	POST /v1/orders/validate	Prima chiamata che va fatta per poter vendete un TdV.
	POST /v1/orders/issue	Emette un TdV.
	POST /v1/guest/orders/issue	Emette un TdV guest.
	POST /v1/redeemableOrders/issue	Emette un TdV da retail.
Redemption controller	POST /v1/tickets/{pnr}/redemption/validate	Metodo che si occupa di validare un PNR che verrà riscattato, sia per TdV anonimi sia non anonimi.
	POST /v1/tickets/{pnr}/redeem	Associa un PNR ad un userID.
	POST /v1/tickets/{pnr}/guest/redeem	Associa un PNR ad un deviceID per TdV guest.
Refund controller	PATCH /v1/Tickets/{pnr}/refund	Storna l'emissione di un PNR avvenuta con successo.
Validation controller	POST /v1/tickets/{pnr}/activate	Per l'attivazione di un TdV.

	POST /v1/tickets/{pnr}/validate	Per inviare i log di autovalidazione, nel caso di mobile validation.
	GET /v1/tickets/{pnr}/validations	Per restituire le autovalidazioni e attivazioni, nel caso di mobile validation.
Migrations controller	PATCH/v1/tickets/{pnr}/changeDevice	Permette di spostare un titolo attivo su un nuovo device.
Signatures controller	GET /v1/staticSignatures/certificates	Per ottenere le chiavi pubbliche per la validazione della firma del QRcode.
	POST /v1/qrcode/verifySignatures	Permette di verificare la firma di un QRcode.

Tabella 8 - Metodi per pre-pagato (biglietti)

CATALOG CONTROLLER

GET /v1/catalog

La catalog è l'interfaccia che permette ad un OT di ottenere tutti i ticket acquistabili dal CSR-D. Le informazioni che vengono restituite forniscono una fotografia aggiornata al momento della chiamata e possono quindi essere salvate sui sistemi degli OT, evitando la necessità di interrogare l'API in tempo reale per ogni richiesta. Questo approccio consente di adottare politiche di caching tali da rendere i due sistemi autonomi. È comunque raccomandato mantenere un allineamento periodico, ad esempio con aggiornamenti giornalieri, o in corrispondenza di eventi particolari come variazioni tariffarie. Sarà onere dell'OT verificare i nuovi dati ottenuti dalla catalog e salvarli sul proprio sistema di Ticketing.

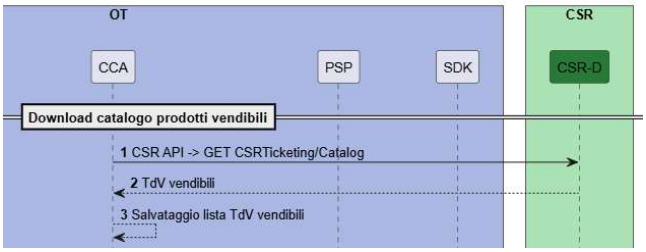


Figura 14 - Sequence diagram per Catalog

All'interno della catalog, oltre ai dati basilari (codice articolo, descrizioni, ecc.) si segnalano i seguenti campi particolari:

- enviromentType
  - Tpl: significa che il TdV è vendibile tramite userId;
  - TplGuest: significa che il TdV è vendibile anche in maniera anonima;
- Validity
  - SingleRideValidity: titolo di corsa singola;
  - MultiRideValidity: titolo multicorsa, con numero di corse disponibili inserito nel campo rides;
  - DurationValidity: titolo a tempo con validità definita tramite i campi unit e value;
  - CustomValidity: validità customizzabile per sviluppi futuri;
- ValidityArea: rappresenta la zona (regione, provincia, comune) di copertura del TdV.

I titoli facenti parte del perimetro CSR-D sono titoli di viaggio a durata (DurationValidty) o in alternativa multicorsa (MultiRideValidity).

A titolo esemplificativo l'App degli OT dovrà gestire la visualizzazione degli articoli come segue:

- *visualizzare solo gli item con proprietà environmentType=Tpl nel caso di acquisto titolo da loggato.*
- *visualizzare solo gli item con environmentType=Tpl_Guest nel caso di acquisto titolo guest.*

CONFIRMATION CONTROLLER

PATCH /v1/orders/{orderId}/confirm

Il metodo serve per confermare un determinato orderId. La response attesa in caso di esito positivo per questo metodo è un 202. Successivamente si dovrà richiamare il metodo Status per ottenere lo stato aggiornato dell'ordine specifico.

In caso di esito non positivo, la conferma dell'ordine va ritentata con politiche di retry finché non si ottiene un 202.

GET/v1/orders/{orderId}/status

Il metodo Status serve per identificare lo stato di un orderId. Quando il CSR-D effettuerà tutte le operazioni di conferma e convalida genererà il PNR, che sarà quindi ritornato all'interno dell'oggetto confirmationResults. Il metodo gestisce tre stati differenti:

- Pending: significa che la conferma è ancora in corso. L'OT dovrà prevedere politiche di polling per poter ottenere lo stato definitivo.
- Confirmed: significa che l'orderId è stato confermato e pertanto sono stati generati i PNR.
- NotConfirmed: significa che l'orderId non è stato confermato per problemi tecnici e pertanto l'OT dovrà effettuare una nuova emissione.

ORDERS CONTROLLER

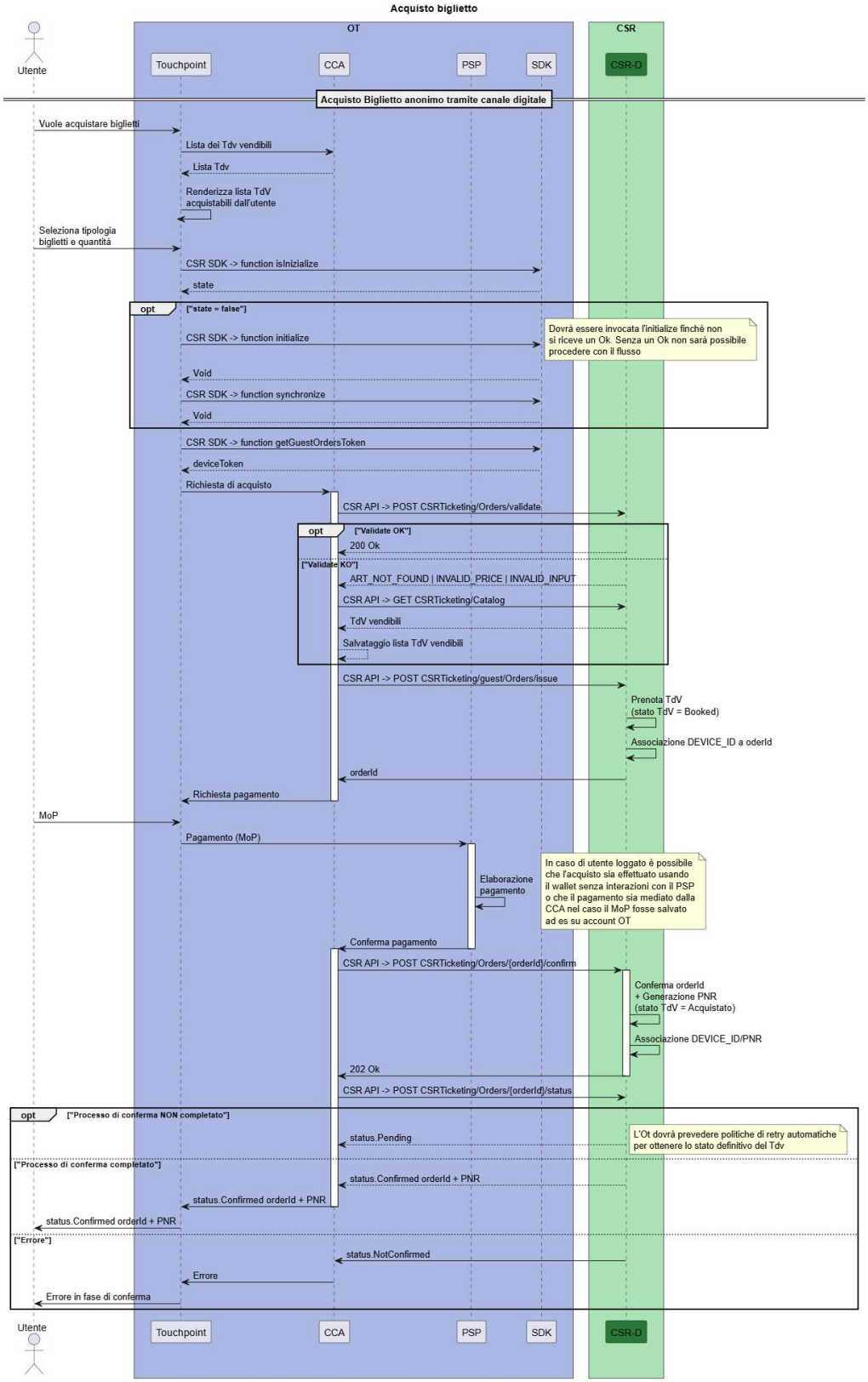


Figura 15 - Processo di acquisto ticket non guest

POST /v1/orders/validate

La validate è la prima chiamata che va fatta per poter vendere un TdV sia in modalità guest che in modalità autenticata. Obiettivo della validate è quello di certificare che il TdV sia vendibile e che tutti i parametri siano corretti; pertanto, si suggerisce di inserire tale chiamata all'interno del flusso di acquisto nell'app degli OT (es. all'interno del carrello). In caso di errore del metodo validate l'OT potrà invocare nuovamente la catalog per poter aggiornare la politica tariffaria.

POST /v1/orders/issue

Il metodo ha la funzione di emettere un TdV, lasciandolo in uno stato "Booked". Per eseguire correttamente l'operazione è necessario fornire una serie di campi obbligatori, quali**:**

- userId
- tickets.code
- tickets.serviceTypeId
- tickets.enviromentId
- tickets.quantity
- tickets.unitPrice
- Tickets.issueParameter (valorizzato a 'None')

Si specifica che i campi ticket.userData (es. nome e cognome) sono da compilare in maniera obbligatoria. In caso di errore dovrà essere necessario effettuare una nuova emissione; il CSR-D non accetta l'emissione parziale di un orderId. Il CSR-D associa ciascuna emissione fatta allo userId che l'ha effettuata.

POST /v1/guest/orders/issue

Il metodo ha la funzione di emettere un TdV guest, lasciandolo in uno stato "Booked". Per eseguire correttamente l'operazione è necessario fornire una serie di campi obbligatori, quali:

- deviceToken
- tickets.code
- tickets.serviceTypeId
- tickets.enviromentId
- tickets.quantity
- tickets.unitPrice
- Tickets.issueParameter (valorizzato a 'None')

Si specifica che il campo email dell'oggetto UserData è da compilare in maniera obbligatoria. Per quanto riguarda il campo deviceToken, questo rappresenta il device firmato presso il quale si sta effettuando l'operazione di acquisto del TdV. Il valore deve essere recuperato tramite il metodo getGuestOrdersToken specifico dell'SDK. In caso di errore del metodo dovrà essere necessario effettuare una nuova emissione; il CSR-D non accetta l'emissione parziale di un orderId. Il CSR-D effettua le emissioni di titoli "anonimi" associando il TdV al deviceID, in quanto non è presente lo userId.

POST /v1/redeemableOrders/issue

Il metodo permette di emettere uno o più TdV che saranno poi oggetto di riscatto in app tramite l'associazione del TdV (PNR + ticketCode) ad un account / deviceId.

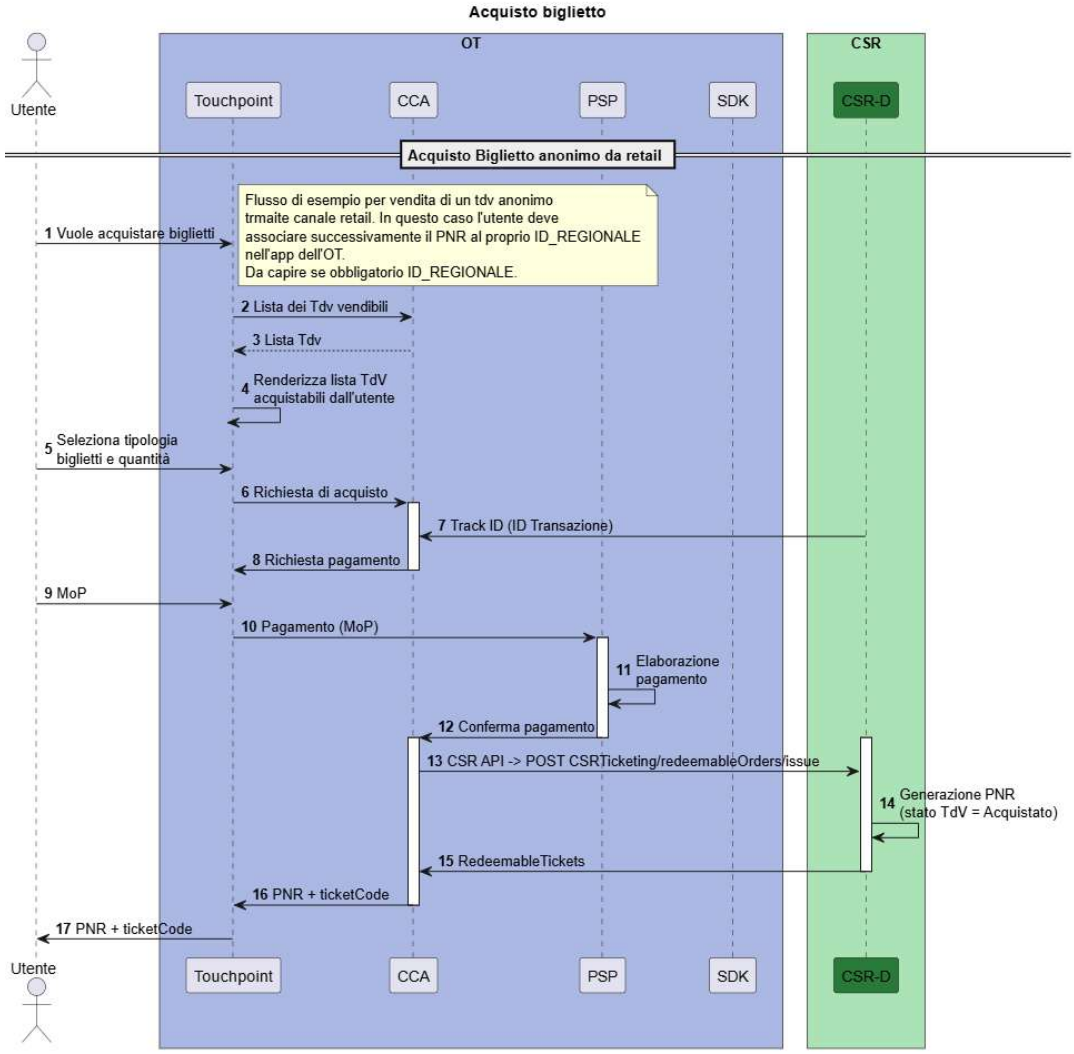


Figura 16 - Processo di acquisto titolo da retail per associazione PNR

REDEMPTION CONTROLLER

Questo controller si occupa della gestione dei PNR emessi da retail che devono essere “riscattati” su flusso dedicato. Il TdV può essere riscattato secondo due modalità, ossia collegando il TdV ad uno userId (per TdV non guest) oppure ad un deviceId (per TdV guest). Il flusso si compone di due step:

1. Verifica del PNR: il metodo “validate” effettua una verifica formale che il PNR sia riscattabile;
2. Riscatto del PNR:
   - Guest/redeem: effettua il riscatto di un PNR guest, utilizzando il deviceToken;
   - Redeem: effettua il riscatto di un PNR, comunicando lo userId.

Di seguito è riportato il dettaglio del metodo e del flusso.

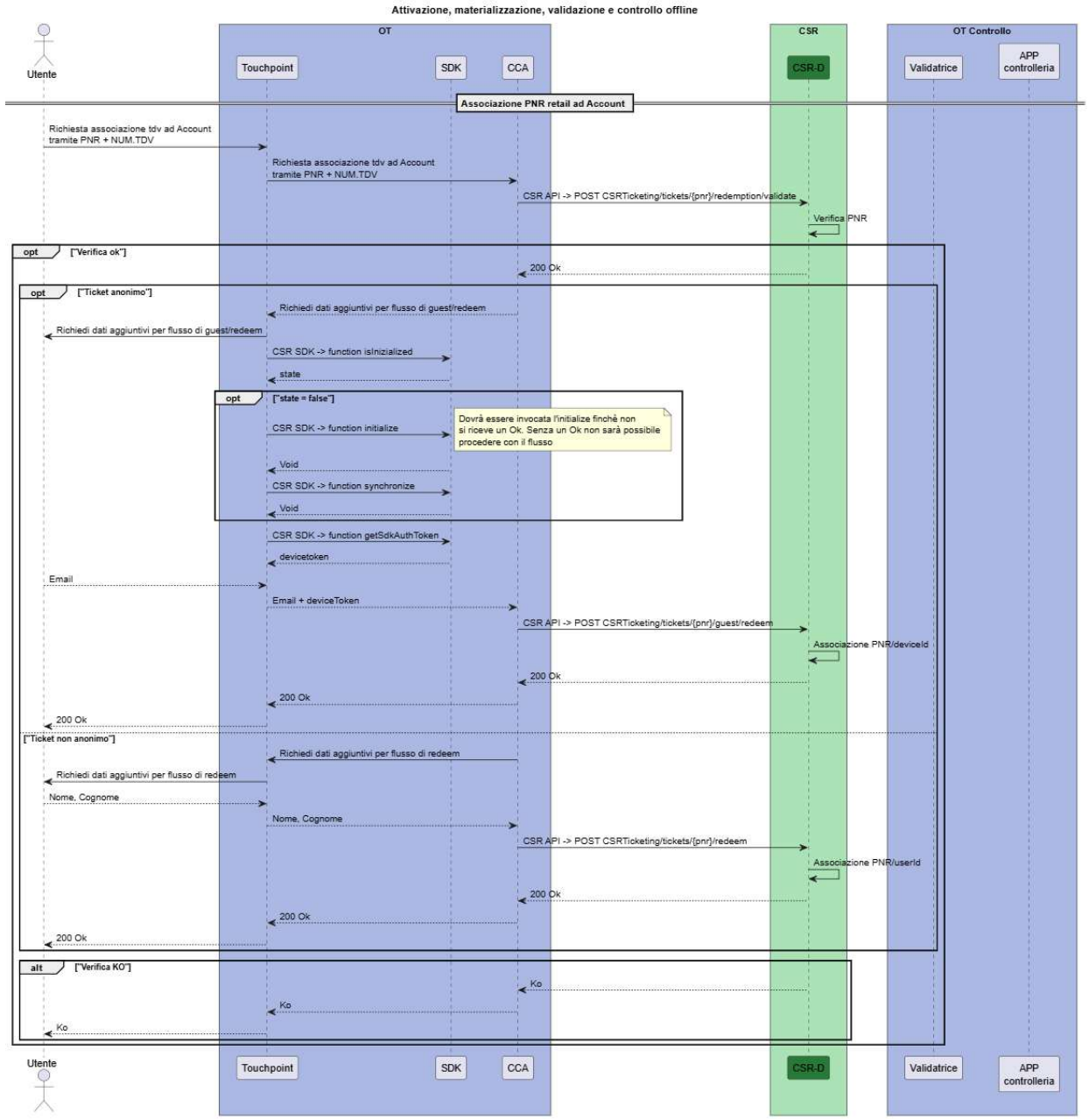


Figura 17 - Processo di riscatto PNR

POST /v1/tickets/{pnr}/redemption/validate

Prima di procedere con un riscatto è disponibile un metodo che si occupa di validare il PNR che si andrà a riscattare, il quale esegue dei controlli per verificare che il PNR sia riscattabile. La validate deve essere invocata sia per TdV guest che TdV non guest. I dati da passare in input sono PNR e TicketCode. Il doppio parametro è richiesto ai fini della sicurezza e di evitare all'utente il riscatto di un titolo differente dal suo a causa della digitazione errata del proprio PNR.

POST /v1/tickets/{pnr}/redeem

Il metodo si occupa di associare un PNR ad uno userID. Per poter associare un PNR è necessario inserire il PNR e un secondo fattore di sicurezza, il ticketCode. Tale processo è valido solo per i biglietti, in quanto gli abbonamenti sono collegati alla tessera regionale, e pertanto sono automaticamente associati ad un identificativo; in quel caso, per gli abbonamenti, va infatti seguito

il flusso di associazione e attivazione della tessera regionale a un account, come descritto nel capitolo precedente dedicato agli abbonamenti.

POST /v1/tickets/{pnr}/guest/redeem

Il metodo si occupa di associare un PNR ad un deviceId. Per poter associare un PNR è necessario inserire PNR e un secondo fattore di sicurezza, il ticketCode. Per quanto riguarda il campo deviceToken, questo rappresenta il device firmato presso il quale si sta effettuando l'operazione di acquisto del TdV. Il valore deve essere recuperato tramite il metodo getTicketToken specifico dell'SDK.

REFUND CONTROLLER

PATCH /v1/Tickets/{pnr}/refund

Il metodo si occupa di stornare l'emissione andata a buon fine rispetto ad un PNR (biglietto). Nel caso in cui l'utente abbia acquistato più biglietti nello stesso orderId, il metodo di storno va invocato per ciascun PNR emesso e per i soli PNR da stornare. Lo storno sarà abilitato solo dallo stesso OT che ha effettuato la vendita.

VALIDATION CONTROLLER

Ai fini della gestione degli eventi di validazione di un titolo, sia che esso sia un abbonamento o un biglietto, saranno resi disponibili tre metodi (API):

- **POST /v1/tickets/{pnr}/activate**: permette l'attivazione di un titolo, modificandone lo stato (passa dallo stato non attivo a stato attivo) e calcolando le date di validità (da questo momento in poi inizia la validità del TdV);
- **POST /v1/tickets/{pnr}/validate**: permette la registrazione degli eventi di validazione di un determinato titolo;
- **GET /v1/tickets/{pnr}/validations**: permette il recupero dell'elenco delle attivazioni e degli eventi di validazione di un determinato titolo.

POST /v1/tickets/{pnr}/activate

Il metodo activate si occupa di effettuare l'attivazione di un TdV all'interno del CSR-D. Senza questa chiamata, e senza una risposta positiva, il titolo rimane nello stato di non attivo e quindi non viene generato né restituito il QRcode associato, indispensabile ai fini della validazione. Per poter utilizzare il metodo è fondamentale aver generato un deviceToken. Nel caso in cui l'utente sia loggato è possibile passare in input anche lo **userId**, così da associare l'attivazione direttamente all'utente stesso. Si possono passare opzionalmente i valori relativi al numero di corse (per la multi-attivazione sui carnet) ed alle coordinate geografiche.

Il metodo restituisce, in caso di esito positivo e nel caso in cui siano state passate anche le coordinate geografiche, un valore "isInsideArea" che informa l'OT se il TdV sia stato validato all'interno dell'area di riferimento del TdV oppure al di fuori di essa.

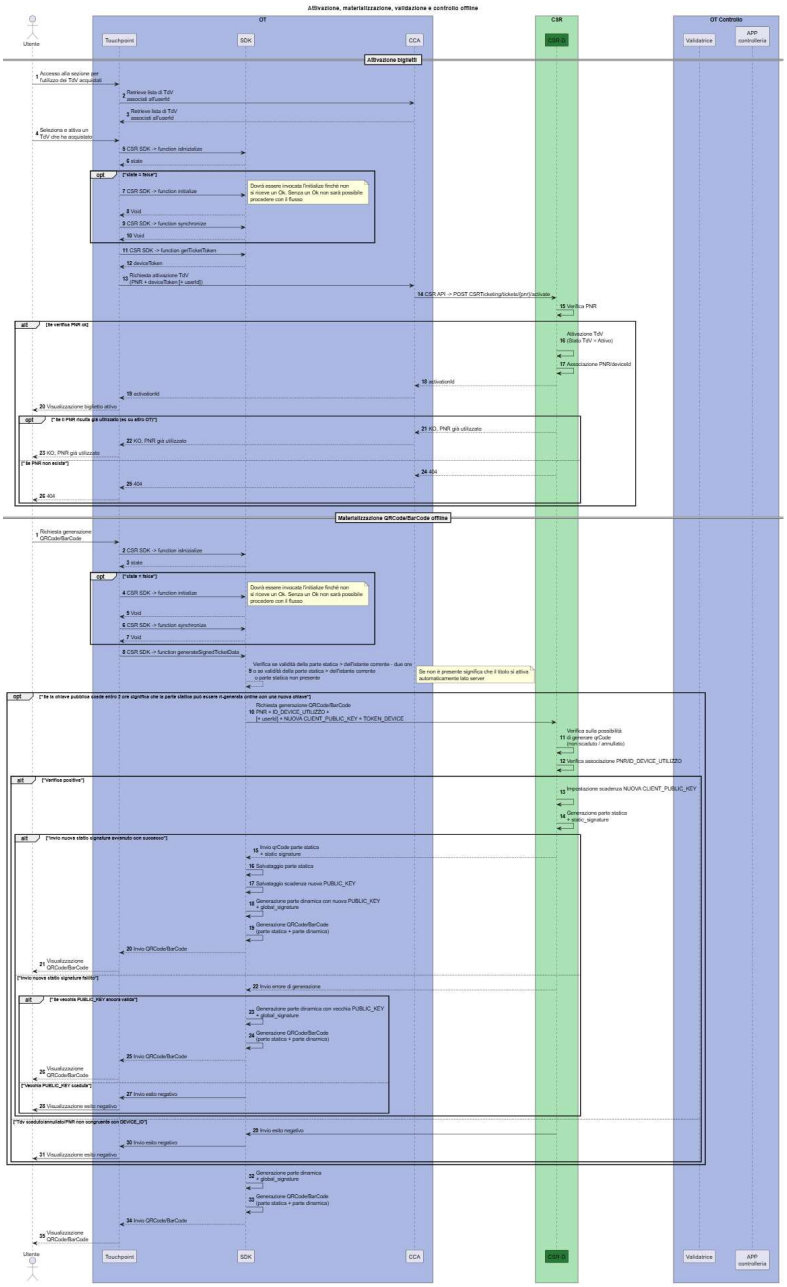


Figura 18 – Flusso di attivazione TdV e generazione QRcode

POST /v1/tickets/{pnr}/validate

Il metodo serve per inviare il log di auto-validazione al CSR-D effettuato tramite le mobile application degli OT. Qualora venisse passata anche la posizione geografica (oggetto coordinate.latitude e coordinate.longitude) il CSR-D effettuerà un controllo per verificare se la tipologia del TdV (es. Mi1-Mi3) sia conforme con la posizione dell'utente; in caso di esito positivo verrà restituito il metodo isInsideArea = true, altrimenti false. Per quanto riguarda il campo deviceToken, questo rappresenta il device firmato presso il quale si sta effettuando l'operazione di acquisto del TdV. Il valore deve essere recuperato tramite il metodo getTicketToken specifico dell'SDK. Le validazioni possono essere inviate al CSR-D in tempo reale oppure a posteriori, ma comunque entro il mese operativo, così da permettere al CSR-D stesso di elaborare le opportune reportistiche di clearing.

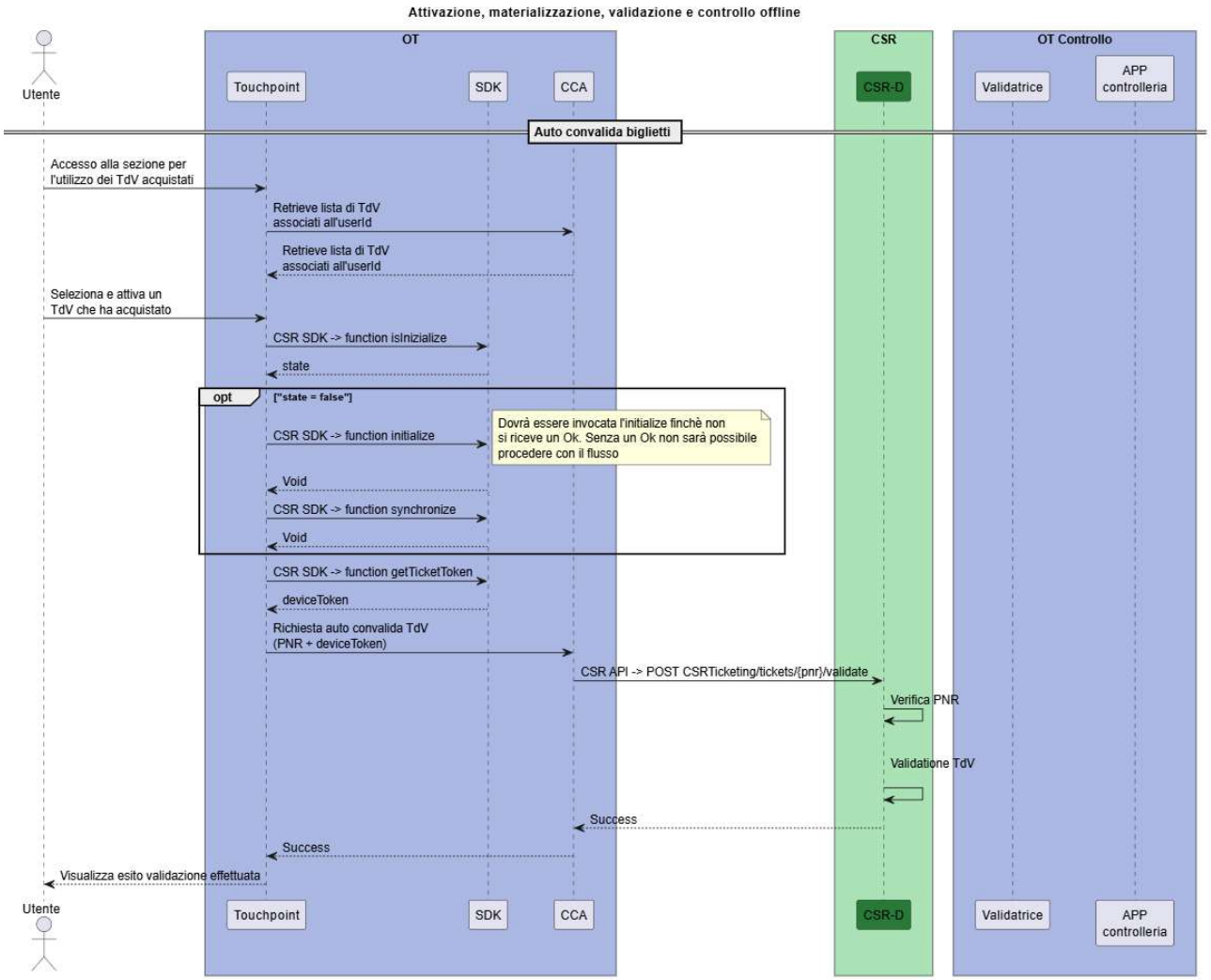


Figura 19 – Flusso di auto-convalida TdV

GET /v1/tickets/{pnr}/validations

Il metodo restituisce, dato un TdV (PNR), tutte le auto-validazioni e le attivazioni effettuate con tale TdV. Il metodo restituisce di default, per questioni di performance, gli ultimi 30 giorni di validazioni. Qualora si volesse ottenere dati per un periodo temporale differente, si possono usare due campi in input, quali searchValidationsFrom e searchValidationsUntil per delimitare il periodo temporale di ricerca, a patto che la differenza tra le due date non superi i 30 giorni.

MIGRATION CONTROLLER

Per trasferire un titolo attivo su un nuovo device occorre utilizzare la **PATCH /v1/tickets/{pnr}/changeDevice** di seguito descritta.

PATCH /v1/Tickets/{pnr}/changeDevice

Il metodo change device consente a un utente che ha cambiato dispositivo di associare i propri **TdV** al nuovo device, scegliendo i titoli da migrare presenti nella sezione personale dell'app dell'OT. Per fare questo passaggio dovrà invocare questo metodo passando in input il deviceToken e lo userId. Per quanto riguarda il campo deviceToken, questo rappresenta il device firmato presso il quale si

sta effettuando l'operazione di acquisto del TdV. Il valore deve essere recuperato tramite il metodo getTicketToken specifico dell'SDK.

Nel CSR-D possono essere configurati tre parametri:

- Numero massimo di migrazioni giornaliere per userId;
- Numero massimo di migrazioni mensili per userId;
- Tempo minimo tra una migrazione e l'altra.

Tali parametri potranno essere definiti in fase di configurazione del sistema.

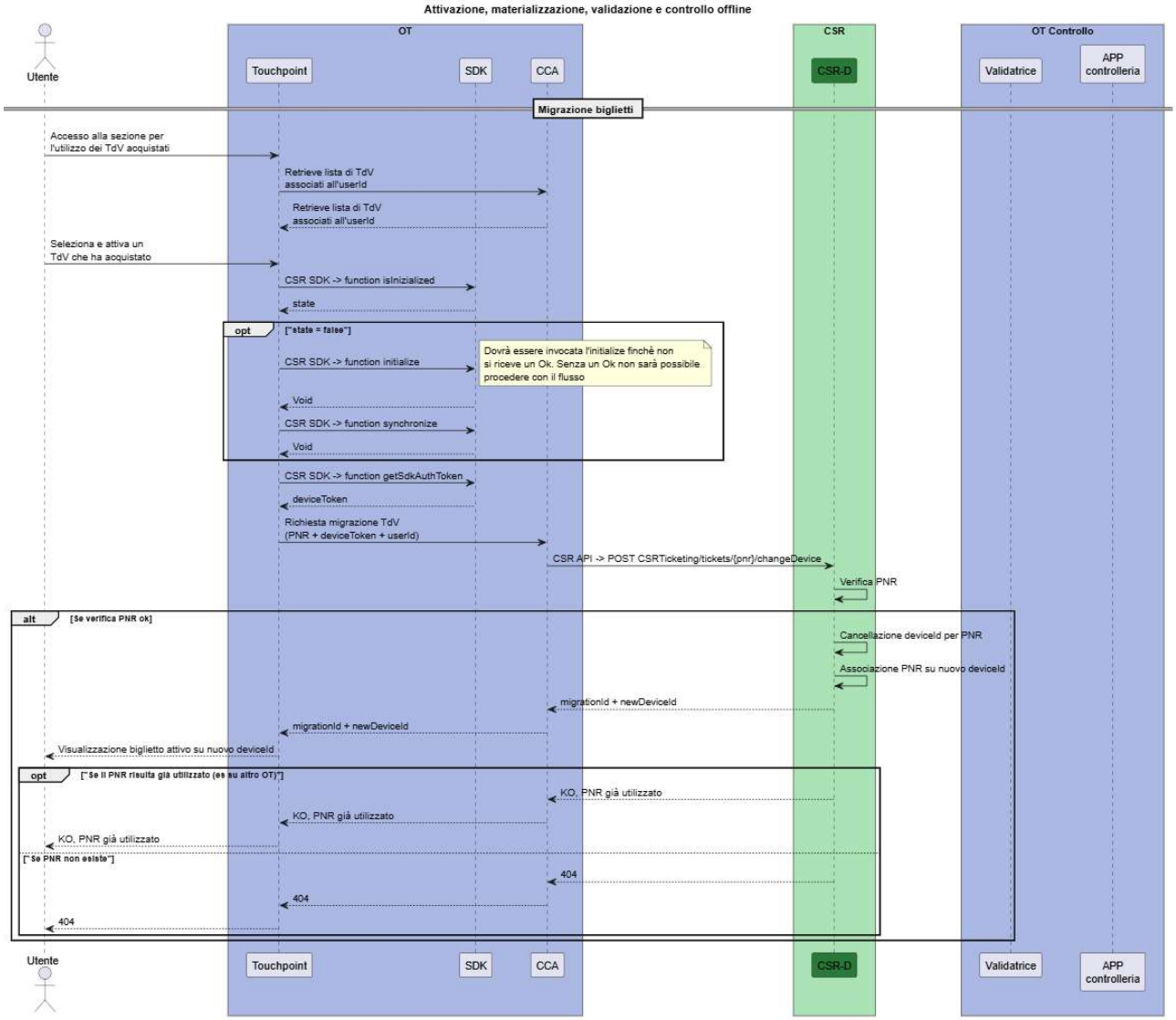


Figura 20 - Processo di migrazione deviceId

SIGNATURES CONTROLLER

GET /v1/staticSignatures/certificates

Il metodo permette alle CCA di ottenere le chiavi pubbliche utili ai fini della validazione della firma del QRcode. Con la chiave pubblica si potrà verificare che il QRcode sia autentico e che non sia stato manipolato da soggetti terzi. Come descritto nella figura sottostante, l'OT riceve dal CSR-D la chiave di sicurezza pubblica del server e dovrà distribuirla sugli apparati di validazione.

L'algoritmo di decodifica e di parsing del QRcode sarà inserito in allegato una volta definite tutte le specifiche di sicurezza. Per identificare l'allegato si veda il riferimento alla Tabella 4 - Riferimenti alle specifiche tecniche.

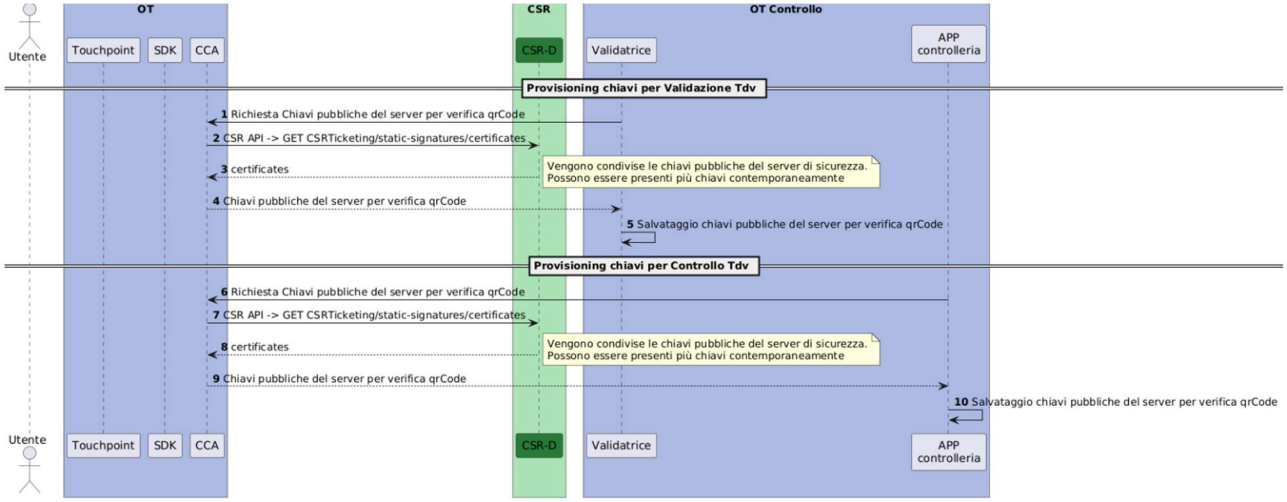


Figura 21 - Processo di condivisione delle chiavi pubbliche utili ai fini della validazione e controllo

POST /v1/qrcode/verifySignatures

Il metodo permette di verificare la firma di un QRcode passato come parametro in input. Tale metodo restituisce due oggetti:

- isValidServerSignature: booleano che determina se la firma della parte statica del QRcode è valida;
- isValidClientSignature: booleano che determina se la firma della parte dinamica generata dal client del QRcode è valida.

Se il metodo risponde con true per entrambe le chiavi, l'OT dovrà verificare il contenuto del QRcode tramite le indicazioni presenti ai capitoli 4 e 5 del documento *ARIA_CSR Digitale_SpecificheQRcodeDinamico.*

4.3. Servizi post-pagato

I servizi nell'ambito del post-pagato consentono di gestire in maniera completa tutti i processi legati ai titoli digitali acquistati in modalità post-pagata. Gli Operatori possono interagire con il sistema attraverso diverse API, organizzate in specifici controller dedicati a ciascuna area funzionale:

- L'**Environment Controller,** per gestire i parametri tecnici necessari all'interazione con il sistema CSR-D;
- Il **Validation Controller**, per notificare la generazione e l'utilizzo degli Alias di pagamento per la gestione del post-pagato;
- Il **Payment Controller**, per gestire gli esiti dei pagamenti, eventuali storni ed effettuare le verifiche sugli Alias.

Ciascun controller espone API dedicate che permettono di accedere alle funzionalità specifiche del modulo, consentendo agli Operatori di integrare in modo completo e modulare i processi post-pagati all'interno delle loro applicazioni.

Ai fini di una miglior comprensione, i controller e i metodi ad essi associati sono riportati di seguito in formato tabellare, corredati da una breve descrizione di ciascuna API.

Controller	Metodi	Descrizione
Environment controller	GET /v1/Environment/GetEnvironmentSettings	Permette di ottenere un parametro tecnico da utilizzare per le chiamate successive. Tale parametro è documentato nei vari swagger come campo obbligatorio denominato "Setting" presente nell'header delle api.
Validation controller	POST /v1/Validation/SavePaymentAlias	Notifica la creazione di un nuovo Alias di pagamento dal TSP al CSR-D.
	POST /v1/Validation/SendValidations	Invia un evento di validazione (tap) associato a un Alias di pagamento.
	POST /v1/Validation/BestFare	Il metodo consente di calcolare runtime la tariffa di un determinato spostamento a seguito di una condivisione di un set minimo di dati.
Payment controller	GET /v1/Payment/GetPayments	Restituisce un file con i pagamenti da effettuare, preautorizzazioni da annullare e tap fuori zona.
	POST /v1/Payment/SavePaymentResults	Invia gli esiti dei pagamenti effettuati.
	POST /v1/Payment/RefundPayment	Richiede lo storno di un pagamento associato a un TrackId.
	GET /v1/Payment/GetDenyPayments	Elenco degli Alias con almeno un pagamento fallito in un intervallo di date.
	GET /v1/Payment/VerifyPayment	Verifica se un Alias ha pagamenti insoluti (Trackid, data, importo).
Componente TSP	POST /Tokenize	Tokenizza una carta di debito/credito (sia fisica che virtuale) durante un pagamento.

Tabella 9 - Metodi per il post-pagato

ENVIRONMENT CONTROLLER

GET /v1/Environment/GetEnvironmentSettings

Il metodo GET /v1/Environment/GetEnvironmentSettings deve essere invocato come prima chiamata del flusso di integrazione e restituisce il parametro tecnico denominato Setting, obbligatorio nell'header di tutte le chiamate successive. Il campo Setting è una stringa che concatena tramite punto e virgola tre identificativi:

- TipoInterfaccia;
- NomePortale;
- IdSistema.

Un esempio rappresentativo del valore è il seguente: MOMO;ARIACLIENTEMV;0001.
La response include inoltre i campi CompanyCode, CompanyName e BusinessName, riferiti all'operatore autenticato.
Il parametro ottenuto sarà poi statico e potrà essere salvato in cache, evitando così di dover ripetere la chiamata al metodo ad ogni richiesta.

Esempio

```
{
  "EnvironmentSettings":
  [
    {
      "Setting": "MOMO;ARIACLIENTEMV;0001",
      "CompanyCode": "0001",
      "CompanyName": "Ragione Sociale",
      "BusinessName ": "Ragione Sociale Commerciale",
    }
  ]
}
```

VALIDATION CONTROLLER

POST/v1/Validation/SavePaymentAlias

Il metodo permette di notificare dal modulo TSP al modulo CSR-D la generazione di un nuovo paymentAlias. **Tale metodo non dovrà essere integrato dagli OT in quanto è parte della tratta TSP e CSR-D.** Per ciascuna richiesta vengono trasmessi:

- companyId: identificativo univoco dell'OT;
- dateExpiration: definita da parametro di sistema.

Il valore paymentAliasPrec è invece un parametro che riporta il paymentAlias ed è valorizzato solo nel caso in cui la stessa carta si ripresenti entro x giorni (dove x è un parametro di sistema) successivi alla scadenza del token. In questa logica, poiché il motore di best fare elabora le richieste solo dopo

48h operative, è possibile riconciliare le validazioni (“tappate”) effettuate con la stessa carta ma associate a paymentAlias differenti.

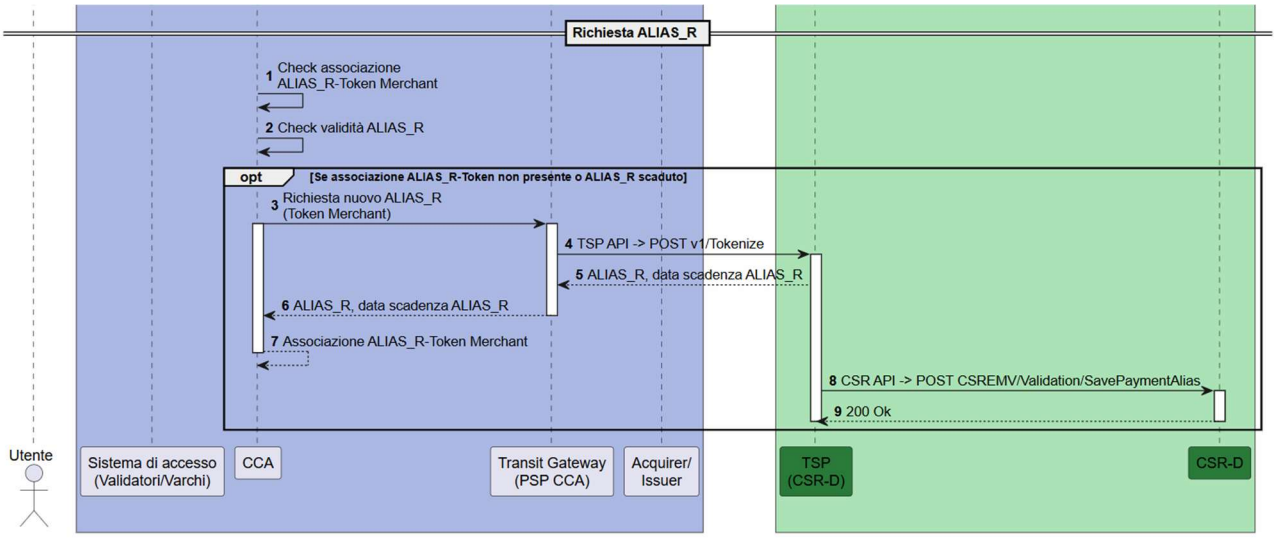


Figura 22 - Sequence diagram per notificare al modulo del CSR-D che è stato generato un nuovo Alias

POST /v1/Validation/SendValidations

Il metodo consente di trasmettere, dato un paymentAlias, le informazioni relative agli eventi di validazione. All’interno dell’oggetto ValidationDetail sono presenti i seguenti campi significativi:

- lineID e rideId: da compilare con l’id della linea e l’id della corsa legate all’evento di tap. Questi campi non sono necessari, ad esempio, per i tap in ambito metropolitano o ferroviario, effettuati utilizzando tornelli o validatori di terra;
- physicStopId: da valorizzare con la fermata fisica presso cui è stato effettuato il tap;
- businessStopId: se non disponibile la fermata fisica, si può indicare la fermata commerciale. In tale caso la fermata fisica viene ricavata tramite una tabella di trascodifica che l’OT dovrà condividere con il CSR-D.

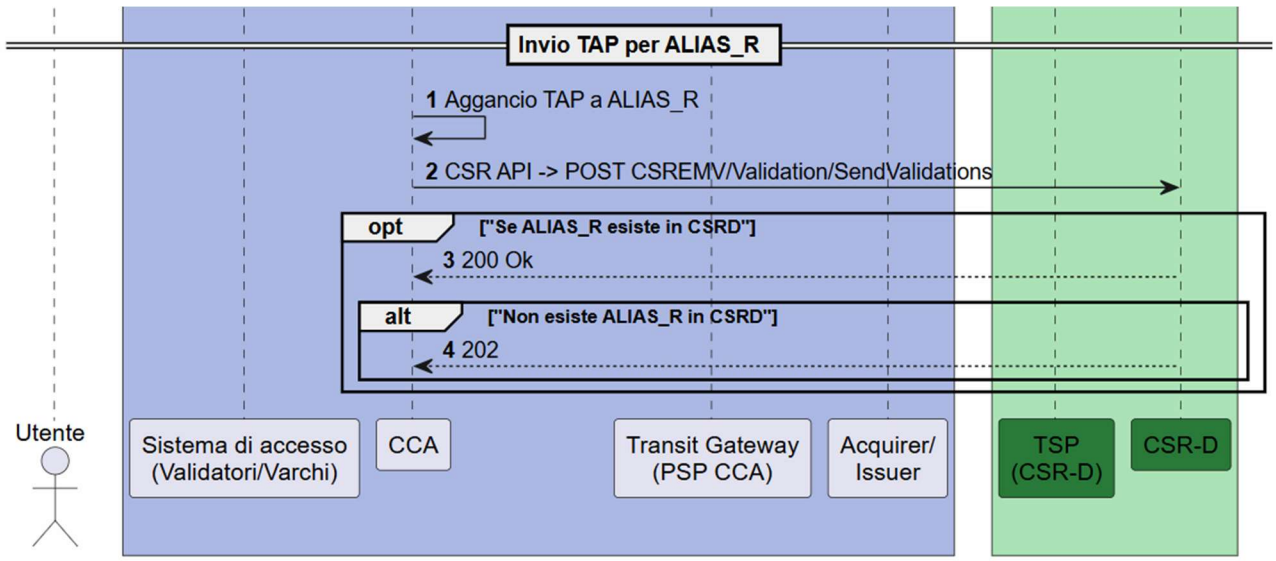


Figura 23 - Sequence diagram per inviare un evento di tap

Il metodo restituisce un codice 202 qualora il paymentAlias non sia ancora stato registrato nel CSR-D, nel caso in cui il TSP, per i tempi di elaborazione, non abbia ancora inviato un nuovo Alias al CSR-D. In questa casistica il CSR-D riceverà successivamente la certificazione dell'Alias generato dal TSP, per cui l'OT non si dovrà preoccupare di inviare nuovamente il tap. Inoltre, nel processo di GetPayments, sarà restituita anche una proprietà (paymentAliasConfirmed), che indentifica se l'Alias è stato certificato dal TSP.

HTTP Status Code	Codice di errore	Messaggio	Descrizione
400	**INVALID_INPUT**	Invalid Input	Input non valido
202	**ACCEPTED**	Accepted successful response	PaymentAlias non ancora registrato nel CSR-D.

Tabella 10 – SendValidations Error Code

Esempio


```
{
 "PaymentAlias": "5255551200011",
 "ValidationDetails":
 [
  {
   "TapId": "123ABC",
   "PreAuthCode": "123ABC",
   "LineId": "123ABC",
   "RideId": 123456,
   "PhysicStopId": "123ABC",
   "BusinessStopId": "123ABC",
   "Latitude": 43.672909,
   "Longitude": 13.295525,
   "DateValidation": "2025-08-11T09:22:56.282Z",
   "VehicleId": "123ABC",
   "CheckStatus": "CHECKIN"
  }
 ]
}
```


POST /v1/Validation/BestFare

Il metodo risponde ad una richiesta specifica di un operatore e non rientra tra le API che gli operatori dovranno necessariamente sviluppare per la corretta integrazione con la piattaforma CSR-D.

Il metodo consente di calcolare il valore della tariffa in funzione dei dati che sono stati inseriti in input, sulla base quindi di origine e destinazione dello spostamento. Il metodo è stateless, pertanto non tiene in considerazione richieste precedenti ma si basa sempre e solo sull'input ricevuto (lista di validazioni effettuate da una certa azienda sulla rete di trasporto). L'OT riceve nella risposta la tariffa attribuita come descritto nell'"Esempio risposta" di seguito riportato e nello swagger.

Esempio richiesta


```
{
  "validationDetails":
 [

   {

     "networkId": "string",

     "lineId": "123ABC",

     "rideId": "123456",

     "physicStopId": "123ABC",

     "businessStopId": "123ABC",

      "latitude": 43.67291,

      "longitude": 13.295525,

      "dateValidation": "2025-08-11T09:22:56.282Z",

      "vehicleId": "124",

      "checkStatus": "CHECKIN"
    }
  ]
 }
```


Esempio risposta


```
{

  "bestFareId": "4dd12b84-bd49-46d3-91e6-a077b61df9b5",

  "actionType": 1,

  "price": 100.23,
```

```
"issueTicketsList": [
 {
  "articleCode": "ARTDSA",
  "articleDesc": "Descrizione articolo",
  "price": 25.23,
  "fareValidationList": [
   {
    "companyId": "12",
    "companyDesc": "AGI (MI+PV)",
    "tapType": 1,
    "tapDescription": "Zona MI1 - Comune di Milano",
    "lineId": "123ABC",
    "rideId": "123456",
    "physicStopId": "123ABC",
    "businessStopId": "123ABC",
    "latitude": 0,
    "longitude": 13.295525,
    "dateValidation": "2025-08-11T09:22:56.282Z",
    "vehicleId": "124",
    "checkType": "CHECKIN",
```

```
      "checkTypeVirtual": true
    }
   ]
  }
 ],
 "ignoredFareValidationList": [
  {
   "companyId": "12",
   "companyDesc": "AGI (MI+PV)",
   "tapType": 1,
   "tapDescription": "Zona MI1 - Comune di Milano",
   "lineId": "123ABC",
   "rideId": "123456",
   "physicStopId": "123ABC",
   "businessStopId": "123ABC",
   "latitude": 0,
   "longitude": 13.295525,
   "dateValidation": "2025-08-11T09:22:56.282Z",
   "vehicleId": "124",
   "checkType": "CHECKIN",
   "checkTypeVirtual": true
```

```
    }
  ]
}
```

PAYMENT CONTROLLER

Il payment controller è una sezione del CSR-D dedicata alla gestione delle operazioni di ricostruzione dei checkout virtuali grazie all'analisi di tutti i tap ricevuti dai singoli OT, al calcolo della best fare e alla gestione dei pagamenti, compresi eventuali storni.

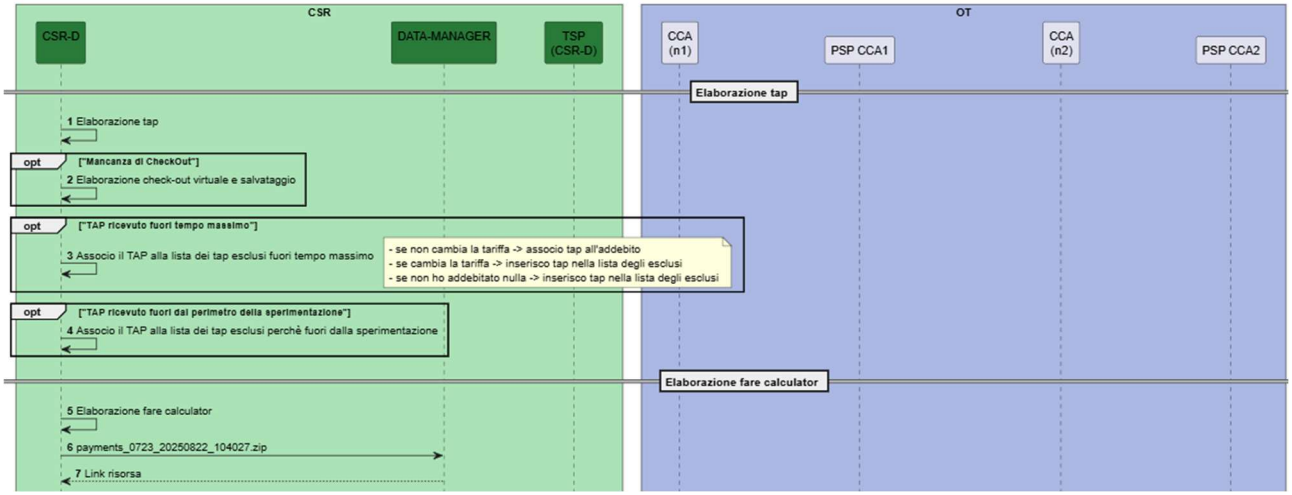


Figura 24 - Sequence diagram per elaborazione tap e calcolo best fare

Sono disponibili due metodi:

- **GetPayments**: restituisce, per una data specificata in fase di richiesta, un file in formato ZIP contenente la lista dei pagamenti da effettuare, la lista delle preauth da annullare e/o l'elenco delle tappate fuori dall'area di sperimentazione o ricevute in ritardo;
- **SavePaymentResults**: riceve in input un file in formato ZIP opportunamente formattato contenente gli esiti dei pagamenti richiesti dal motore di best fare.

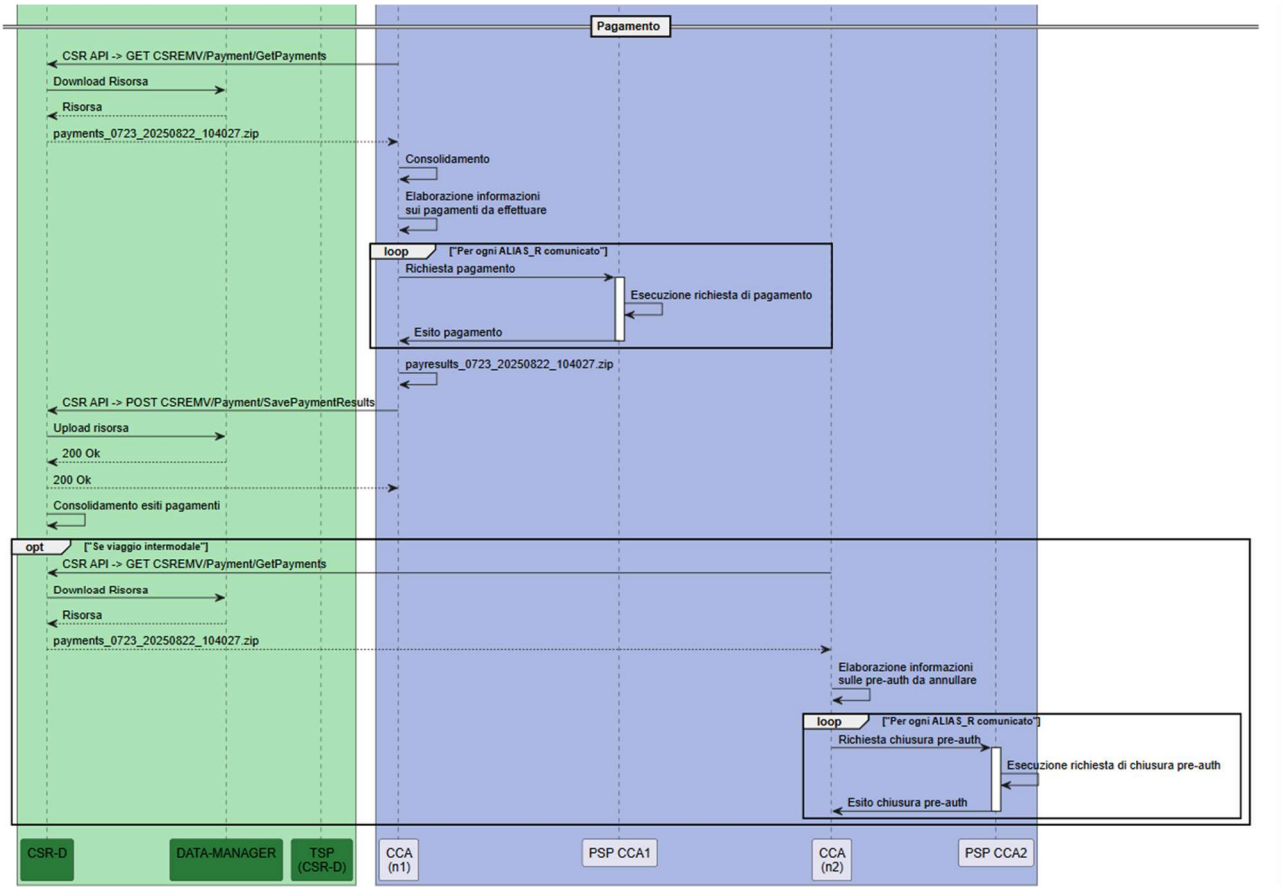


Figura 25 - Sequence diagram per payment controller

GET /v1/Payment/GetPayments

Il metodo accetta in input obbligatoriamente una data, che rappresenta la data di elaborazione della best fare. Ogni OT può richiamare il metodo in qualsiasi momento, scaricando il file relativo alla data desiderata, anche a ritroso di giorni. Il CSR-D manterrà, per questioni di performance, il file già elaborato in una directory del data manager per un periodo di tempo limitato. Qualora la richiesta si riferisca a un periodo troppo distante (es. superiore a 45 giorni), il CSR-D provvede a rielaborare i dati e a rigenerare il file. In questo caso, il servizio restituisce un response code 202, invitando l'OT a richiedere nuovamente il file. Qualora la richiesta venga effettuata in un arco temporale troppo ristretto, ad esempio inferiore al tempo necessario per il calcolo della Best fare, il servizio restituisce un response code 404. Il file con tutti i dettagli è in formato TXT contenente struttura JSON e sarà inserito all'interno di uno ZIP denominato in questo formato:

payments _ @identificativoOT _ yyyymmdd _ hhmmss . zip

Il parametro @identificativoOT è il codice dell'OT univoco presente all'interno del CSR-D. Il nome di un file di esempio è questo: *payments_0723_20250822_104027*.zip. Il file TXT contenente struttura JSON segue lo stesso approccio.

Nome parametro	Tipo	Nullable	Note
FarePaymentList	List<FarePayment>	No	Elenco pagamenti

<FarePayment>

Nome parametro	Tipo	Nullable	Note
PaymentAlias	String	No	È l'Alias regionale a cui imputare il pagamento

ActionType	Int	No	1=pagamento, 2=annullamento preauth, 3=altro
PaymentAliasConfirmed	Boolean	No	Valorizzato a True quando il paymentAlias è confermato dal TSP
TrackId	String	Yes	Identificativo univoco della transazione, valorizzato per i pagamenti, ActionType=1. Ad un TrackId possono essere associati uno o più titoli (TicketId)
Price	Decimal	Yes	L'importo da pagare, valorizzato per i pagamenti, ActionType=1
FareValidationList	List<FareValidation>	No	Elenco id univoci tappate valutate nel best fare
IssueTicketsList	List<IssueTicket>	No	Elenco dei titoli (TicketId) emessi ed associati al TrackId e che hanno prodotto l'importo da pagare
IgnoredFareValidationList"	List<IgnoredValidation>	No	Elenco id univoci tappate scartate dal best fare

<IssueTicket>

Nome parametro	Tipo	Nullable	Note
TicketId	Int	No	Identificativo univoco del titolo emesso
ArticleCode	String	Yes	Codice articolo utilizzato nell'emissione del titolo
ArticleDesc	String	No	Descrizione articolo utilizzato nell'emissione del titolo
Price	Decimal	No	Importo associato al titolo emesso

<FareValidation>

Nome parametro	Tipo	Nullable	Note
TapId	String	No	Identificativo univoco della tappata
PreAuthCode	String	Yes	Individua la pre-auth effettuata quando presente, valorizzato per Action Type = 1 o 2
CompanyId	String	No	Identifica l'azienda dove è stata fatta la tappata
CompanyDesc	String	Yes	Breve descrizione azienda
TapType	Int	No	Valore assegnato in fase di best fare, 1=inclusa nei pagamenti, 2=inclusa nelle preauth da annullare, 3=tappata fuori zona sperimentazione, 4=fuori tempo limite, 5=scartata
TapDescription	String	Yes	Breve descrizione del tipo tappata
LineId	String	No*	Identificativo Linea
RideId	Integer	No*	Identificativo Corsa
PhysicStopId	String	No**	Identificativo Fermata fisica
BusinessStopId	String	No**	Identificativo Fermata commerciale
Latitude	Double	No**	Latitudine
Longitude	Double	No**	Longitudine
DateValidation	String	Yes	Data e ora validazione
VehicleId	String	Yes	Identificativo veicolo
TicketId	Int	Yes	Identificativo univoco del titolo emesso

<IgnoredValidation>

Nome parametro	Tipo	Nullable	Note
TapId	String	No	Identificativo univoco della tappata
PreAuthCode	String	Yes	Individua la pre-auth effettuata quando presente, valorizzato per Action Type = 1 o 2
CompanyId	String	No	Identifica l'azienda dove è stata fatta la tappata
CompanyDesc	String	Yes	Breve descrizione azienda
TapType	Int	No	Valore assegnato in fase di best fare, 1=inclusa nei pagamenti, 2=inclusa nelle preauth da annullare, 3=tappata fuori zona sperimentazione, 4=fuori tempo limite, 5=scartata
TapDescription	String	Yes	Breve descrizione del tipo tappata
LineId	String	No*	Identificativo Linea
RideId	String	No*	Identificativo Corsa
PhysicStopId	String	No**	Identificativo Fermata fisica
BusinessStopId	String	No**	Identificativo Fermata commerciale
Latitude	Double	No**	Latitudine
Longitude	Double	No**	Longitudine
DateValidation	String	Yes	Data e ora validazione
VehicleId	String	Yes	Identificativo veicolo
CheckType	String	Yes	Identificativo della tappata CHECKIN o CHECKOUT
CheckTypeVirtual	Int	Yes	Informazione se la tappata è virtuale (1) oppure no (0)
TicketId	Int	Yes	Identificativo univoco del titolo emesso

Tabella 11 - File JSON GetPayments

HTTP Status Code	Codice di errore	Messaggio	Descrizione
202	**Accepted**	Accepted successful response	Status restituito quando, a fronte di una richiesta, il CSR-D è obbligato a ri-generare il file. L'OT sarà obbligato pertanto a richiedere successivamente, a parità di data, il file da scaricare.
404	**Not found**	Not found	Status restituito quando viene richiesto un file per un periodo di tempo non conforme con il calcolo della best fare.

Tabella 12 – HTTP Status ode GetPayments

Nel caso in cui un OT riceva, per un determinato paymentAlias, un valore sull'oggetto PaymentAliasConfirmed = false significa che la carta non è stata riconosciuta dal TSP e che quest'ultimo non ha potuto generare correttamente l'Alias, ad esempio per dati condivisi errati. In questo caso, l'OT non potrà richiedere un addebito per tale carta. Gli Operatori di Trasporto potranno approfondire la motivazione per cui il TSP non ha generato l'Alias e perché non è stata possibile la tokenizzazione aprendo un ticket di assistenza tramite applicativo di back office.

Esempio


```
{
    "FarePaymentList": [
        {
            "PaymentAlias": "5255551200011",
            "PaymentAliasConfirmed": true,
            "ActionType": 1,
            "TrackId": "123456",
            "Price": 10.5,
            "IssueTicketsList": [
                {
                    "TicketId": 11111,
                    "ArticleCode": "AAAA",
                    "ArticleDesc": "BBBBBBB VVVVVV",
                    "Price": 5.50
                },
                {
                    "TicketId": 22222,
                    "ArticleCode": "XXXX",
                    "ArticleDesc": "PPPPPP AAAAAA",
                    "Price": 3.20
                },
                {..
                }
            ],
            "FareValidationList": [
                {
                    "TapId": "123ABC",
                    "PreAuthCode": "123ABC",
                    "CompanyId": "123",
                    "CompanyDesc": "AZIENDA 123",
                    "TapType": 1,
                    "TapDescription": "DESCRIZIONE TAPTYPE 1",
                    "LineId": "123ABC",
                    "RideId": 123456,
                    "PhysicStopId": "123ABC",
                    "BusinessStopId": "123ABC",
                    "Latitude": 43.672909,
                    "Longitude": 13.295525,
                    "DateValidation": "2025-08-11T09:22:56.282Z",
                    "VehicleId": "123ABC",
                    "TicketId": 1234444,
                },
                {
                    "TapId": "454PPP",
                    "PreAuthCode": "454PPP",
                    "CompanyId": "123",
                    "CompanyDesc": "AZIENDA 123",
                    "TapType": 1,
                    "TapDescription": "DESCRIZIONE TAPTYPE 1",
                    "LineId": "454PPP",
                    "RideId": 123456,
                    "PhysicStopId": "454PPP",
                    "BusinessStopId": "454PPP",
                    "Latitude": 43.672909,
                    "Longitude": 13.295525,
                    "DateValidation": "2025-08-11T09:22:56.282Z",
                    "VehicleId": "454PPP",
                    "TicketId": 2222222,
```

```
                },
                {..
                }
            ],

            "IgnoredFareValidationList": [
              {
          "TapId": "454PPP",
          "PreAuthCode": "454PPP",
          "CompanyId": "123",
          "CompanyDesc": "AZIENDA 123",
          "TapType": 3,
          "TapDescription": "DESCRIZIONE TAP TYPE",
          "LineId": "454PPP",
          "RideId": 123456,
          "PhysicStopId": "454PPP",
          "BusinessStopId": "454PPP",
          "Latitude": 45.617023,
          "Longitude": 9.416877,
          "DateValidation": "2025-08-11T11:22:56.000Z",
          "VehicleId": "454PPP",
          "CheckType": "CHECKOUT",
          "CheckTypeVirtual": 0,
          "TicketId": null
        }
         ]
        },
        {..
        }
    ]
}
```


POST /v1/Payment/SavePaymentResults

Il metodo accetta in input un file in formato ZIP opportunamente formattato. Il file all'interno sarà un file di testo contenente una struttura in formato JSON con i riferimenti dei pagamenti processati dal calcolo della best fare. La nomenclatura segue il seguente pattern:

payresults _ @identificativoOT _ yyyymmdd _ hhmmss . zip

Il parametro @identificativoOT è il codice dell'OT univoco presente all'interno del CSR-D. Il nome di un file di esempio è *payresults_0723_20250822_104027*.zip. Il file in formato TXT contenente struttura JSON segue lo stesso approccio.

Nome parametro	Tipo	Nullable	Note
PaymentResultList	**List< PaymentResult>**	No	Elenco esiti pagamenti

< PaymentResult>

Nome parametro	Tipo	Nullable	Note
PaymentAlias	String	No	È l'Alias regionale a cui imputare il pagamento
PreAuthCode	String	Yes	Individua la pre-auth effettuata, se presente
TrackId	String	No	Identificativo univoco pagamento
PaymentState	String	No	Stato del pagamento (OK/KO)

Tabella 13 - File JSON PaymentResults

Esempio


```
{
 "PaymentResultList":
 [
  {
   "PaymentAlias": "5255551200011",
   "PreAuthCode": "123ABC",
   "TrackId": "123456",
   "PaymentState": "OK"
  }
 ]
}
```


POST /v1/Payment/RefundPayment

Il metodo si occupa di stornare l'emissione andata a buon fine. Lo storno avviene sempre per TrackId, ovvero per la tariffa addebitata; qualora il TrackId fosse legato ad un solo TdV il metodo si occupa di stornare l'emissione associata ad un determinato TicketId, se invece la tariffa fosse derivata da un insieme di TdV e quindi da più TicketId associati, nell'ambito della logica di Best fare, lo storno sarà applicato a tutti i titoli associati a quel TrackId. Il CSR-D effettuerà un controllo per cui solo l'OT che ha venduto il TdV sarà titolato a poter effettuate uno storno.

In caso di esito negativo (response KO) lo storno non sarà effettuato e potrà essere gestito tramite un applicativo di back office.

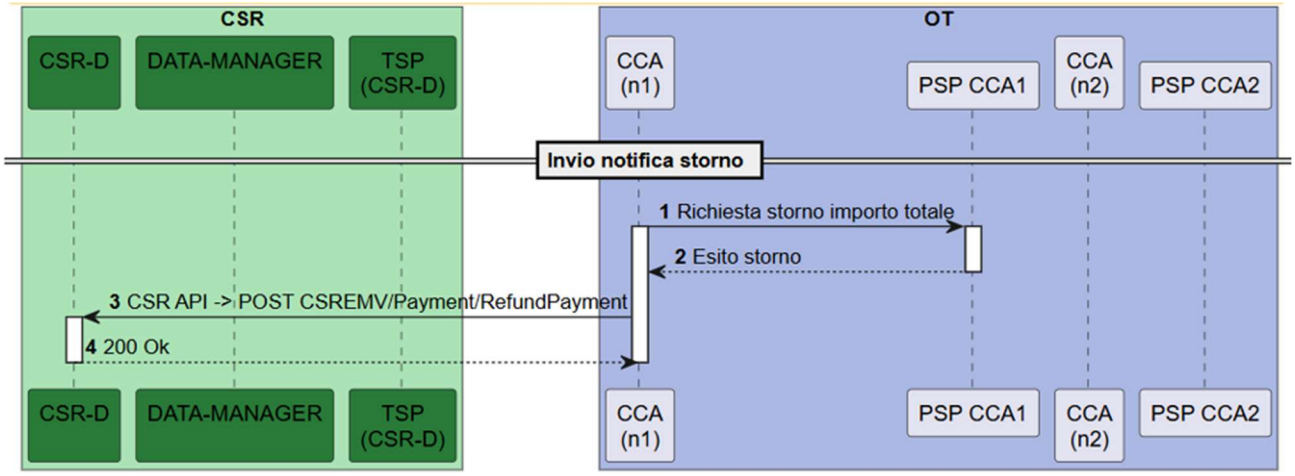


Figura 26 - Sequence diagram per Refund Payment

GET /v1/Payment/GetDenyPayments

Il metodo restituisce gli Alias regionali per i quali è stato registrato almeno un esito di pagamento negativo. La richiesta deve essere effettuata indicando un intervallo temporale (range di date) massimo di due mesi. Per ogni Alias che, nel periodo indicato, presenta almeno un pagamento fallito, saranno esposte le informazioni dell'Alias e tutti i pagamenti falliti (TrackId, data del pagamento, importo e azienda).

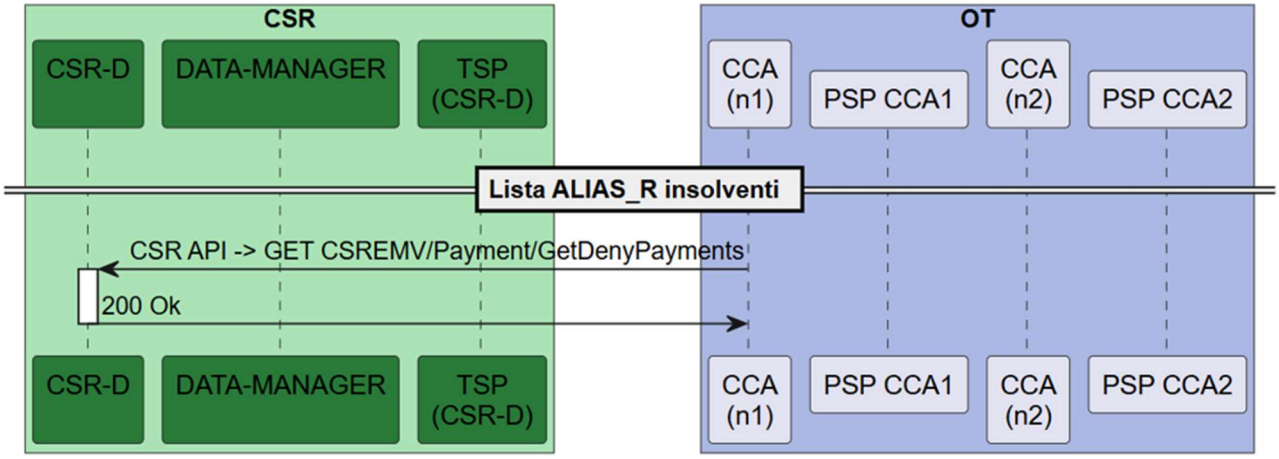


Figura 27 - Sequence diagram per Get Deny Payments

Esempio


```
{
 "denyAliasList":
 [
  {
   "paymentAlias": "5255551200011",
   "denyPaymentList":
   [
    {
     "trackId": "123ABC",
     "paymentDate": "2025-08-11T17:05:00.000Z",
     "price": 10.5,
     "companyId": "123",
     "companyDesc": "AZIENDA 123"
    },
    {..}
   ]
  },
  {..}
 ]
}
```

GET /v1/Payment/VerifyPayment

Il metodo, dato un Alias regionale, restituisce la lista di eventuali pagamenti insolventi (trackId, data del pagamento, importo e azienda).

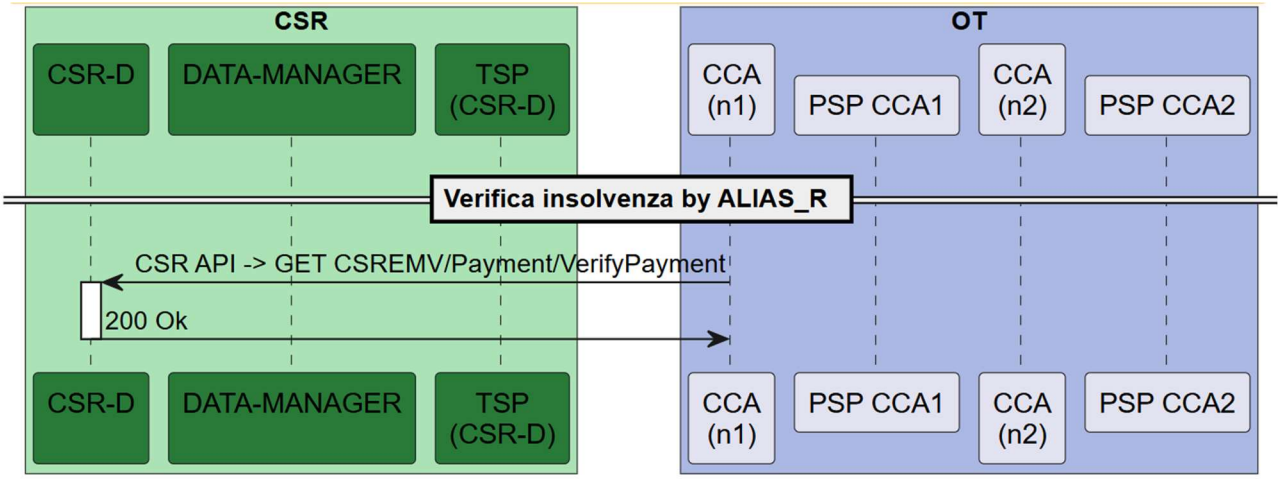


Figura 28 - Sequence diagram per Verify Payment

Esempio


```
{
 "DenyAliasList":
 [
  {
   "PaymentAlias": "5255551200011",
   "DenyPaymentList":
   [
    {
     "trackId": "123ABC",
     "PaymentDate": "2025-08-11T17:05:00.000Z",
     "Amount": 10.5,
     "CompanyId": "123",
     "CompanyDesc": "AZIENDA 123"
    },
    {..}
   ]
  },
  {..}
 ]
}
```

4.3.1. Componente TSP

Il TSP è il modulo del CSR-D che permette di generare un PaymentAlias univoco.

L'introduzione di una componente TSP (Token Service Provider) consente di generare in modo univoco un token di pagamento (Payment Alias) a partire dai dati trasmessi dai PSP degli Operatori di Trasporto.

La disponibilità di un identificativo univoco del metodo di pagamento abilita l'implementazione di logiche commerciali e modelli tariffari indipendenti dal singolo provider o dal transit gateway di riferimento. Tale approccio consente infatti di identificare in modo coerente sia i titoli di viaggio sia le singole tratte, anche quando queste insistono su sistemi od Operatori differenti.

Questa capacità rappresenta un elemento abilitante per l'introduzione di logiche di best fare, permettendo l'aggregazione e l'ottimizzazione tariffaria dei viaggi effettuati dall'utente su più tratte e attraverso differenti sistemi di trasporto.

ARCHITETTURA E INTEGRAZIONI

Il TSP (Tokenizzatore) è un sistema chiuso con l'architettura di seguito rappresentata, composta da:

1. Front-End: Web service di acquisizione delle richieste, valida formalmente i messaggi ricevuti ed esegue i controlli di sicurezza sul mittente del messaggio;
2. Back-End: Componente di gestione delle funzionalità e di accesso alla base dati.

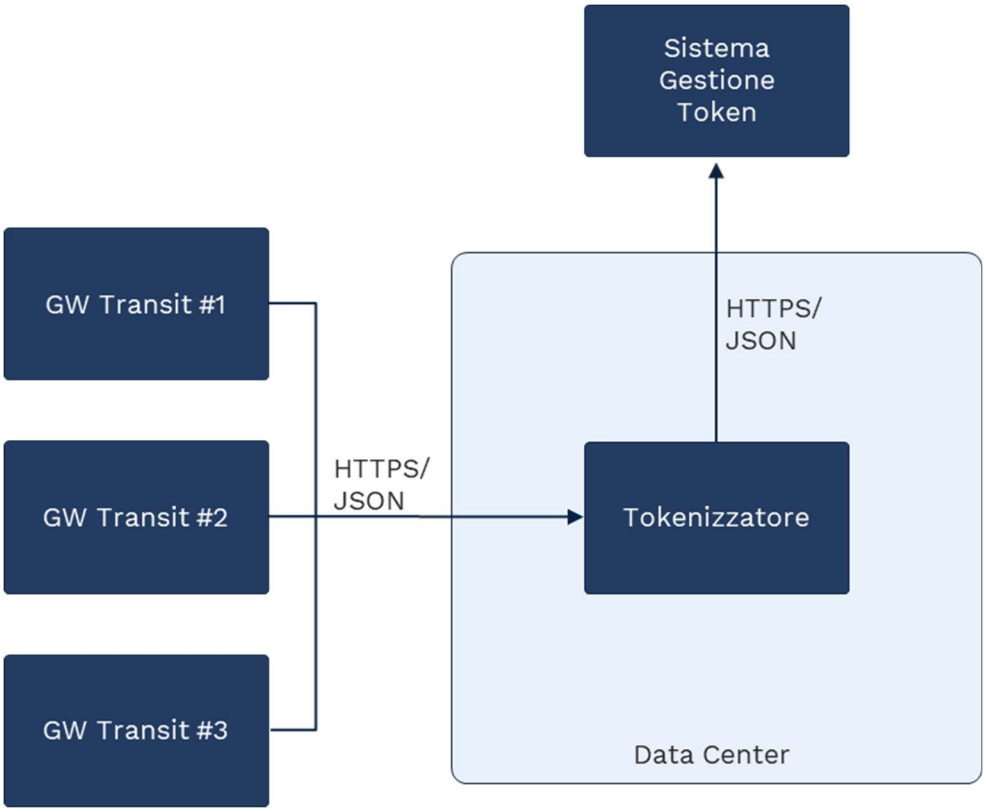


Figura 29 - Architettura del tokenizzatore

Il sistema di Tokenizzazione espone un'interfaccia a microservizi basata su protocollo JSON che espone la seguente funzionalità:

- **Tokenizzazione:** permette la generazione di un Alias (token) associato al titolo di pagamento in modo univoco;

Ciascun messaggio deve essere firmato tramite l'algoritmo HMAC-SHA256 per garantirne l'autenticità e l'integrità.

POST /Tokenize

Questo metodo della componente TSP permette la generazione di un Alias (token) associato univocamente a un titolo di pagamento. L'algoritmo del token prevede che esso sia generato attraverso l'utilizzo di quantità randomiche; non sarà pertanto possibile risalire al titolo di pagamento dal token.

La tokenizzazione di un titolo di pagamento già presente nel sistema restituirà il medesimo token generato in precedenza.

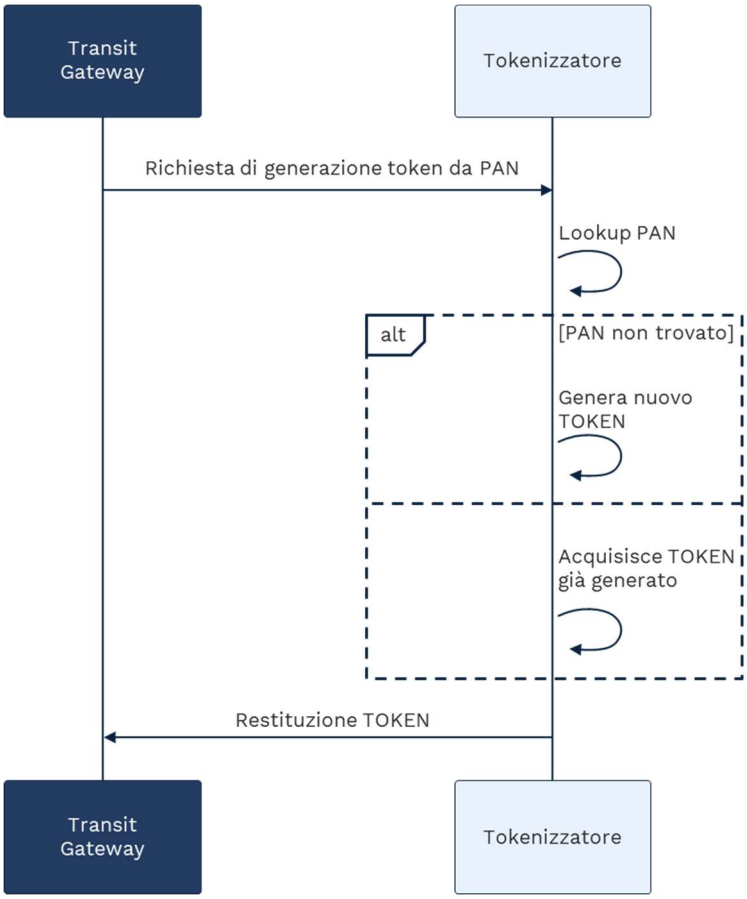


Figura 30 - Tokenizzazione

Messaggio di richiesta

{

"businessUnit": "BU_TEST",

"requestorID": "NETS-ID1",

"requestorUrl": "https://www.netsgroup.com/jpit/enroll/upd",

"requestID": "2639e4dfb-f7d5-4435-b928-f5a42691aea6",

"tokenUniqueRefID": "EN01-IOID0-NN12",

"serviceID": "BILLING",

"encryptedPayLoad": {

"instrumentType": "CARD",

"accountNumber": "5255000000000003",

"accountName": "DEMO",

"expDate": "2612",

"AddInfo": [{ "type": "INFO", "data": [{ "key": "companyId", "value": "ATM", "desc": "Company ID" }] }]

}

}

Risposta positiva del tokenizzatore

{

"requestorID": "NETS-ID1",

"tokenUID": "2323232",

"paymentUID": "324",

"tokenValue": "5255551200011",

"tokenExpiryDate": "2512",

"expiredTokenValue": "5256661200012",

"result": {

"errorCode": "OK",

"errorDetail": "JP000",

"errorDescription": "Operation processed"

}

}

È bene precisare i seguenti punti ai fini di una corretta integrazione del metodo Tokenize:

- I campi BusinessUnit e RequestorID saranno condivisi con i singoli PSP in funzione delle configurazioni e delle abilitazioni;
- Il campo EncryptedPayLoad.InstrumentType va valorizzato sempre con l'attributo CARD;
- Il campo EncryptedPayLoad.AccountName non deve essere valorizzato (non è al momento utilizzato);
- Il campo EncryptedPayLoad.AccountNumber rappresenta il PAN della carta. In caso di carta fisica sarà valorizzato il codice identificativo riportato sulla carta, in caso di carta virtuale sarà valorizzato il codice virtuale generato dal circuito/issuer.

4.4. Validazione e Controlli

Ai fini della gestione delle attività di controlleria e di invio degli esiti derivanti dalla validazione di un TdV da parte dell'utente, vengono messi a disposizione due metodi:

- **/Validation/SendValidationsLog:** permette ai sistemi esterni di inviare il log delle validazioni con esito positivo effettuate dai vari apparati;
- **Check/SendChecksLog:** permette ai sistemi esterni di inviare il log dei controlli.

I metodi esposti dal modulo validazioni e controlli permettono di salvare i dati relativi ai controlli effettuati dai sistemi di controlleria e le validazioni effettuate sui dispositivi di validazione (metro e validatori) o auto-validazione digitale in app.

Controller	Metodi	Descrizione
Environment controller	GET /v1/Environment/GetEnvironmentSettings	Permette di ottenere un parametro tecnico da utilizzare per le chiamate successive. Tale parametro è documentato nei vari swagger come campo obbligatorio denominato "Setting" presente nell'header delle api.
Validation controller	POST /v1/Validation/SendValidationLog	Permette di inviare l'elenco delle validazioni effettuate sui validatori o sui tornelli della metropolitana.
Check controller	POST /v1/Check/SendChecksLog	Permette di inviare l'elenco dei controlli effettuati.

ENVIRONMENT CONTROLLER

GET /v1/Environment/GetEnvironmentSettings

Il primo metodo da invocare è il /Environment/GetEnvironmentSettings, il quale permette di ottenere un parametro tecnico da utilizzare per le chiamate successive. **Il parametro ottenuto sarà poi statico e potrà essere salvato in cache, evitando così di dover ripetere la chiamata al metodo ad ogni richiesta.**

Esempio


```
{
 "EnvironmentSettings":
 [
  {
   "Setting": "MOMO;ARIACLIENTEMV;0001",
   "CompanyCode": "0001",
   "CompanyName": "Ragione Sociale",
   "BusinessName ": "Ragione Sociale Commerciale",
  }
```

```
 ]
}
```

VALIDATION CONTROLLER

POST /v1/Validation/SendValidationsLog

Il metodo consente di trasmettere, dato un PNR, le informazioni relative ad un evento di validazione proveniente dagli apparati di campo. All'interno dell'oggetto validationsLog sono presenti i seguenti campi significativi:

- **LineID e RideId:** da compilare con l'id della linea e l'id della corsa su cui il dispositivo è vestito, ad eccezione di dispositivi installati a terra, come, ad esempio, i tornelli in ambito metropolitano;
- **PhysicStopId:** da valorizzare con la fermata fisica su cui è vestito il dispositivo/tornello;
- **BusinessStopId:** se non disponibile la fermata fisica, si può valorizzare la fermata commerciale. In tal senso la fermata fisica verrà convertita tramite una tabella di trascodifica condivisa con l'OT;

Qualora PhysicStopId o BusinessStopId non vengano valorizzati, sarà necessario passare obbligatoriamente i campi latitude e longitude.

Esempio

```
{
 "ValidationsLog":
 [
  {
   "LineId": "123ABC",
   "RideId": 123456,
   "pnr": "123ABC",
   "PhysicStopId": "123ABC",
   "BusinessStopId": "123ABC",
   "Latitude": 43.672909,
   "Longitude": 13.295525,
   "DateValidation": "2025-08-11T09:22:56.282Z",
   "VehicleId": "123ABC",
   "CheckStatus": "CHECKIN"
  },
  {..}
 ]
}
```

CHECK CONTROLLER

POST /v1/Check/SendChecksLog

Il metodo permette agli OT di inviare l'elenco dei controlli effettuati. L'oggetto CheckLog prevede il PNR, la data del controllo ed uno stato (campo checkResult) che può essere valorizzato in funzione dell'esito del controllo (OK -> CHECKOK | KO -> CHECKKO).

Esempio


```
{
 "ChecksLog":
 [
  {
   "pnr": "123ABC",
   "DateCheck": "2025-08-11T09:22:56.282Z",
   "CheckResult": "CHECKOK",
   "LineId": "123ABC",
   "RideId": 123456,
   "PhysicStopId": "123ABC",
   "BusinessStopId": "123ABC",
   "Latitude": 43.672909,
   "Longitude": 13.295525,
   "VehicleId": "123ABC"
  },
  {..}
 ]
}
```


4.4.1. Step di validazione

Nel corso del paragrafo saranno illustrai gli step di validazione di un TdV.

1. Inizializzazione e recupero chiavi pubbliche server

All'avvio, il validatore o gli apparati di controlleria recuperano la lista aggiornata dei certificati pubblici server tramite l'API GET /v1/staticSignatures/certificates esposta dal sistema centrale.

La lista va salvata localmente e aggiornata secondo le politiche di scadenza dei certificati.

2. Lettura e parsing del QRcode

Lettura del QRcode tramite sistema ottico.

Parsing del contenuto secondo la struttura JSON specificata nel documento ARIA_CSR Digitale_SpecificheQRcodeDinamico (campi da 1 a 16, conversione HEX/ASCII/Epoc).

3. Modalità di validazione delle firme

3.1 Locale (offline) (Validatori)

Verifica firma Client:

- Estrazione del campo firma (pos. 16) e della PubClientkey (pos. 11).
- Verifica della firma digitale del client su tutto il messaggio (esclusa la firma stessa) tramite algoritmo ECDSA secp256r1.

Esito atteso: TRUE.

Verifica firma Server:

- Estrazione del campo firmaServer (pos. 13), indiceChiave (pos. 12).
- Selezione del certificato pubblico con stesso indiceChiave dalla lista locale (recuperata al punto 1).
- Verifica della firma server sulla parte statica del QRcode tramite ECDS secp256r1.

Esito atteso: TRUE.

Se la chiamata al server non va a buon fine, fallback automatico alla modalità offline.

4. Controlli di coerenza e validità dei dati all'interno del QRcode ai fini della validazione

Controllo della parte dinamica:

Verifica che la data/ora corrente sia compresa tra inizioValiditaQr (pos. 15) e fineValiditaQr (pos. 14).

Controllo della parte statica:

- Verifica che la chiave pubblica non sia scaduta (fineValiditaClientkey).
- Verifica che la validità offline (fineValiditaOffline) non sia superata.
- Verifica che la data/ora corrente sia compresa tra dataOraInizioValidita (pos. 7) e dataOraFineValidita (pos. 8).
- Verifica che i dati tariffari (CodArticolo, Partenza, Destinazione) siano adeguati al viaggio dell'utente (logica demandata all'OT in base alle proprie politiche di validazione oline/offline).

4.1 Controlli di coerenza e validità dei dati all'interno del QRcode ai fini del controllo

Controllo della parte dinamica:

Verifica che la data/ora corrente sia compresa tra inizioValiditaQr (pos. 15) e fineValiditaQr (pos. 14).

Controllo della parte statica:

- Verifica che la chiave pubblica non sia scaduta (fineValiditaClientkey).

- Verifica che la validità offline (fineValiditaOffline) non sia superata.
- Verifica che la data/ora corrente sia compresa tra dataOraInizioValidita (pos. 7) e dataOraFineValidita (pos. 8).
- Verifica dal PNR che sia stato effettuato un evento di validazione. L'evento di validazione è presente sul sistema dell'OT pertanto rimane compito dell'OT strutturare il processo di validazione verificando l'effettiva presenza dell'evento sul proprio sistema di validazione/controlleria.

5. Logging e tracciamento

- Registrazione del risultato della validazione (esito, timestamp, PNR, device, operatore).
- Invio del log di validazione (POST /v1/Validation/SendValidations) al sistema centrale tramite endpoint dedicato, con possibilità di invio massivo.
- Invio del log di controllo (POST /v1/Validation/SendValidationLog) al sistema centrale tramite endpoint dedicato, con possibilità di invio massivo.

4.5. SDK

Il CSR-D mette a disposizione un modulo SDK (nativo per IOS e per Android) che ha il compito di generare il QRcode compatibile con il sistema regionale. All'interno dell'SDK sono mantenute tutte le chiavi di sicurezza e gli algoritmi di generazione; pertanto, diventa un punto focale per la sicurezza dei TdV. L'OT riceve dal CSR-D una chiave di sicurezza pubblica del server e dovrà distribuirla sugli apparati di validazione. L'SDK gestirà le logiche di materializzazione e attivazione del TdV, esso consentirà di delegare le logiche di sicurezza legate alla gestione delle chiavi crittografiche (pubblica e privata) utilizzate dal client per la generazione, la validazione locale e la presentazione sicura del barcode dinamico. L'SDK permetterà inoltre la generazione dei QRcode e della coppia di chiavi crittografiche.

La generazione del barcode avverrà in modalità "near-online". Di seguito si riporta il flusso di generazione:

1. L'app OT, anche tramite SDK, genera e gestisce una coppia di chiavi, pubblica e privata del client. La coppia di chiavi verrà salvata nel keystore dell'app e avrà una validità massima di 15 giorni;
2. L'utente, dopo aver acquistato il TdV digitale, lo attiva generando il barcode (Aztec code);
3. L'app dell'OT, anche mediante l'SDK integrata, invia al CSR-D il PNR e la chiave pubblica del client per la generazione della parte statica del barcode;
4. Il CSR-D produce la parte statica del payload, incorporando la chiave pubblica del client e firmandola con la chiave privata del server attraverso una "static signature". La coppia di chiavi (pubblica e privata) del server ha una validità massima di 13 mesi;
5. L'app dell'OT, mediante SDK, riceve la parte statica e genera la parte dinamica firmando globalmente il documento con la chiave privata del client. L'app OT può generare anche offline la parte dinamica del documento fino a che la parte statica è valida. Sarà definita una validità temporale della parte statica, superata la quale l'app OT deve richiedere nuovamente la generazione della parte statica al CSR-D;
6. Il validatore o il controllore eseguono la verifica del barcode tramite la chiave pubblica del server. Devono inoltre controllare la validità del client ed effettuare una verifica temporale sulla validità del barcode mostrato. La chiave pubblica del server viene trasmessa dal CSR-D alle CCA, che la distribuiscono successivamente ai validatori e alle applicazioni di controllo.

Nella cartella SDK, sono contenute le istruzioni di dettaglio per l'integrazione SDK e la relativa documentazione tecnica *(rif. Documento ARIA_CSR Digitale_SDK Integration Guide).*

L'algoritmo per la decodifica del QRcode è descritto nell'allegato *ARIA_CSR Digitale_SpecificheQRcodeDinamico*. Si riportano a titolo esemplificativo il contenuto di un QRcode IVOL e STIBM.

Esempio QRcode IVOL ServiceTypeId > 0

4352535630352D2D415249415F5452454E5F46433245394231352D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D31323530336A0AD5656A0D772B6A0C27326A14101D04E43DFBC99F1E417F6BCBAA2A098149C64A0EFEC9E38AFFF6AB39CC3F5685C91F4B2FC39B612DF6035C18E03C4E9F02FA68ABAEBCBC0EA291D6B87CF4724B289D413031314817838F2B16B9D99CBF22CE08B0F73085C08E0952ADF0ABE426FB0B1EE0ACF1A84C70AFEB4DFACBEEEAC6C914DEF91F9580E3F03E4BF5851CA0EF4703DED96A0AF21B6A0AF1FD3044022013338EF4A2BA9

59FFC977536480ABBC9B8B4D03C19887C625B9F7AF611FA2B6E02201FC7588B92EF1630EB78A7C3C15857636FB4ABE9E41FAFF21EB595E21D51AD58


```
 "qrDetails": {
   "prefix": "CSR",
   "qrVer": "V05",
   "pnr": "RL_TREN_FC2E9B15",
   "partenza": "",
   "destinazione": "",
   "codArticolo": "12503",
   "dataOraInizioValidita": 1779094885,
   "dataOraFineValidita": 1779267371,
   "fineValiditaOffline": 1779181362,
   "fineValiditaClientkey": 1779699741,
   "pubClientkey":
"04E43DFBC99F1E417F6BCBAA2A098149C64A0EFEC9E38AFFF6AB39CC3F5685C91F4B2FC39B612DF6035C18E
03C4E9F02FA68ABAEBCBC0EA291D6B87CF4724B289D",
   "indiceChiave": "A01",
   "firmaServer":
"314817838F2B16B9D99CBF22CE08B0F73085C08E0952ADF0ABE426FB0B1EE0ACF1A84C70AFEB4DFACBEEEA
C6C914DEF91F9580E3F03E4BF5851CA0EF4703DED9",
   "fineValiditaQr": 1779102235,
   "inizioValiditaQr": 1779102205,
   "firma":
"3044022013338EF4A2BA959FFC977536480ABBC9B8B4D03C19887C625B9F7AF611FA2B6E02201FC7588B92EF1
630EB78A7C3C15857636FB4ABE9E41FAFF21EB595E21D51AD58"
 }
```


Esempio QRcode STIBM ServiceTypeId = 0


```
 "qrDetails": {
   "prefix": "CSR",
   "qrVer": "V05",
   "pnr": "RL_TREN_DD9B848A",
   "partenza": "MI1",
```

```
  "destinazione": "MI4",

  "codArticolo": "8750",

  "dataOraInizioValidita": 1779109873,

  "dataOraFineValidita": 1780278299,

  "fineValiditaOffline": 1779196329,

  "fineValiditaClientkey": 1779714715,

  "pubClientkey":
"044110C1C473A1D9900822A669A8D85EE7E622CD54F75847E6DCC068FAE4116B7C6AE5EC448B332C8BA02978
C07CC55CE18BA05D9A258E8CBBB4BB21FC5539DDA9",

  "indiceChiave": "A01",

  "firmaServer":
"C901EE68AD04F6FAF665394F507BBE5E7F42C63455F18A51F113A9028881E3E8195AE881443C01F5B0662BB791
8AB34155A1FEDBF3A081EDA7EDF22CF6B63E69",

  "fineValiditaQr": 1779117286,

  "inizioValiditaQr": 1779117256,

  "firma":
"3046022100CFE7F62266E22A64181E63CF6E131F14FC470EAF05FE0D39D420B4A8E9818B170221009DD39A2C0
692BF7D3382211881862502ACF11BFE94B15646CB66AF16698582B9"

 }
```


Il CSR-D, tramite l'SDK, metterà a disposizione dei metodi interni per poter generare i DeviceToken necessari alle CCA dell'OT per invocare i metodi B2B del CSR-D.Il deviceToken è un JWT firmato che contiene le seguenti informazioni:

Header:

- Alg: ES256
- Typ: JWT

Kid: idenficativo univoco del CSR-D

Payload:

- Iat: DataGenerazione
- Exp: DataScadenza
- DeviceId: DeviceId

Esempio:

eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6Ikhhc2hQdWJsaWNLZXkifQ.eyJpYXQiOjE3NjE
5MTg4NTIsImV4cCI6MTc2MTkxODg1MiwiRGV2aWNlSWQiOiJEZXZpY2VJZCIsImp0aSI6ImExM
mEtMTJhIn0.fEqBHFWIGh4UtKSFmLDbgul72ufAzh_lu3dzLuTArAxt54TVNtu6QoVP0Xv57LgoF0h
zi8xW5sn1geUQs30qiQ

I metodi esposti dal CSR-D quando riceveranno un DeviceToken in request:

- Verificano la firma;
- Verifica che il token sia ancora valido;
- Recupereranno il DeviceId per poter eseguire le operazioni dispositive;

Le logiche di generazione del DeviceToken sono gestite centralmente sia da un punto di vista di logiche operative che di sicurezza (chiavi e validità). Questo permette pertanto una gestione semplice e sicura per rendere il sistema resiliente. La data di scadenza del deviceToken coincide con la data di scadenza della chiave pubblica associata alla chiave privata utilizzata per la firma del token stesso. La coppia di chiavi (pubblica/privata) ha una validità di 7 giorni a partire dalla data della sua prima generazione.