**Narrative Flow – Retrieving the Prescription List**

1. **Client Request Initiation**
   * The client application begins by calling **POST /prescriptions/search** with a body containing the relevant membershipResourceIds, along with required query parameters startDate and endDate.
   * If the membershipResourceIds array is empty, no filtering is applied; all prescriptions visible to the logged-in user are considered.
2. **Prescription Service Call**
   * The **CVS Prescription Service** receives the request and delegates data retrieval to the **Claims Service**, invoking **GET /pharmacyclaims**.
3. **Claims Service – Cache Check**
   * The Claims Service first checks its cache for the requested pharmacy claims data, using a cache key constructed from:
   * claims-v1 + accountIdentifier.idSource~accountIdentifier.idValue
   * + membershipResourceId + startDate + endDate
   * If results are found in the cache, they are returned immediately to the CVS Prescription Service, which transforms them into the **prescription response**.
4. **Claims Service – Token & Authorization Retrieval (if cache miss)**
   * If no cache hit occurs, the Claims Service validates the membership ID from the request.
   * It then calls the **CVS Auth Service** via **GET /cvs\_auth\_token\_data** to obtain an access token.
   * In parallel, three supporting calls are made:
     1. **Core Proxy Service** (/userauthorizations) → Retrieves authorization rules for the requester.
     2. **Member Profile Service** (/familyaccessrules) → Fetches family access rules for dependent members.
     3. **CVS Backend** (/getClaimsHistory) → Retrieves raw claims data for all plan members visible to the logged-in user, using the token from CVS Auth.
5. **Claims Service – Filtering & Rule Enforcement**
   * The raw claims results are refined:
     1. **Family Access Rules** are applied (e.g., only return eligible dependent data).
     2. **UAF Rules (privacy restrictions)** are applied (e.g., exclude sensitive drugs or HIPAA-restricted dependents).
   * Claims are then de-duplicated: if multiple claims share the same uniqueRxId, the claim with the most recent fillDate is retained.
6. **Claims Service – Response Assembly & Caching**
   * A response object is created using the defined mapping logic (e.g., drug.name → drugName, fillDate → lastFilledDate, dispensedQuantity → quantity).
   * This object is written into cache for future retrieval using the same key structure.
   * The filtered and transformed results are returned to the CVS Prescription Service.
7. **Prescription Service – Final Transformation**
   * The CVS Prescription Service applies final filtering rules:
     1. If the request body contains membershipResourceIds, only claims matching those IDs are transformed into prescriptions.
     2. If the request body has none, all claims are returned.
   * The service maps claims into the **prescription response format**, including fields such as prescriptionId, drugName, drugStrength, quantity, lastFilledDate, daysSupply, fulfilledBy, and prescriberName.
8. **Response to Client**
   * The CVS Prescription Service returns the final prescription list to the client.
   * If required parameters (startDate, endDate) are missing, the service returns a **400 Bad Request**.
   * If no prescriptions are found, a **200 response with an empty list** is sent.

✅ **In short:**  
The client posts a search request → Prescription Service delegates to Claims Service → Claims Service checks cache, otherwise fetches data via CVS Auth, Core Proxy, Member Profile, and CVS Backend → Applies access and privacy rules → Builds and caches response → Prescription Service transforms and filters → Returns the prescription list to the client.

**Example Response from POST /prescriptions/search**

[

{

"membershipResourceId": "5~263801696+31+1+20180101+788678+C+3",

"memberFirstName": "John",

"memberLastName": "Doe",

"relationshipToSubscriber": "Self",

"startDate": "2019-04-01",

"endDate": "2019-12-05",

"prescriptionList": [

{

"claimId": "200023611694001",

"prescriptionId": "674529584",

"drugName": "Lipitor",

"drugStrength": "200mg",

"quantity": 15,

"lastFilledDate": "2018-04-12",

"daysSupply": 5,

"fulfilledBy": "Caremark prescription service",

"prescriberName": "John Doe"

}

]

}

]

This shows the structure of the returned payload:

* **Top-level attributes** → membership identifiers, member info, and the search date range.
* **prescriptionList array** → individual prescriptions, each with identifiers (claimId, prescriptionId), drug details (drugName, drugStrength), supply/quantity, last fill date, prescriber, and fulfillment source.

Do you want me to also pull out the **detailed field mapping table** (how raw claim fields like .memberClaims[].drug.name map into this response JSON)? That’s included in the document as well.

**Prescription List (POST /prescriptions/search) — Mapping**

| **Claims API Field** | **Prescription List Response Field** | **Transform / Notes** |
| --- | --- | --- |
| .details[].memberClaims[].claimNumber | [].prescriptionList[].claimId | Direct copy. |
| .details[].memberClaims[].uniqueRxId | [].prescriptionList[].prescriptionId | Direct copy. |
| .details[].memberClaims[].drug.name | [].prescriptionList[].drugName | Title case. |
| .details[].memberClaims[].drug.strength | [].prescriptionList[].drugStrength | String → number. |
| .details[].memberClaims[].daysSupplyQuantity | [].prescriptionList[].daysSupply | String → number. |
| .details[].memberClaims[].dispensedQuantity | [].prescriptionList[].quantity | String → number. |
| .details[].memberFirstName | [].memberFirstName | Direct copy. |
| .details[].memberLastName | [].memberLastName | Direct copy. |
| .details[].memberClaims[].fillDate | [].prescriptionList[].lastFilledDate | If multiple claims share uniqueRxId, keep the one with **most recent** fillDate. |
| .details[].memberClaims[].prescriberName | [].prescriptionList[].prescriberName | Direct copy. |
| .details[].memberClaims[].pharmacyName | [].prescriptionList[].fulfilledBy | Direct copy. |
| .details[].relationshipToSubscriber | [].relationshipToSubscriber | Direct copy. |
| .details[].membershipResourceId | [].membershipResourceId | Direct copy. |
| prescriptionsRequest.startDate | [].startDate | From request. |
| prescriptionsRequest.endDate | [].endDate | From request. |

**Dedup + selection rule for list:** “If… more than 1 claim object share the same uniqueRxId, select the claim with the most recent fillDate.”

**Prescription Detail (GET /prescriptions?claimId=…) — Mapping**

| **Claims API Field** | **Prescription Detail Response Field** | **Transform / Notes** |
| --- | --- | --- |
| .details[].memberClaims[].claimNumber | .claimId | Direct copy. |
| .details[].memberClaims[].uniqueRxId | .prescriptionId | Direct copy. |
| .details[].memberClaims[].fillDate | .lastFilledDate | Direct copy. |
| .details[].memberFirstName | .memberFirstName | Direct copy. |
| .details[].memberLastName | .memberLastName | Direct copy. |
| .details[].relationshipToSubscriber | .relationshipToSubscriber | Direct copy. |
| .details[].membershipResourceId | .membershipResourceId | Direct copy. |
| .details[].memberClaims[].drug.name | .drugName | Title case. |
| .details[].memberClaims[].drug.strength | .drugStrength | Direct copy. |
| .details[].memberClaims[].drug.dosageForm | .drugForm | Title case. |
| .details[].memberClaims[].daysSupplyQuantity | .daysSupply | String → number. |
| .details[].memberClaims[].dispensedQuantity | .quantity | String → number. |
| .details[].memberClaims[].refillsLeft | .refillsLeft | String → number. |
| .details[].memberClaims[].prescriber.fullName | .prescriberName | Title case. |
| .details[].memberClaims[].pharmacyName | .fulfilledBy | Title case. |
| *Derived* (.memberClaims[].payAmount + .memberClaims[].clientPayAmount) | .estimatedCost | **Calculated sum** of payAmount and clientPayAmount. |
| *(none / optional)* | .prescriptionExpirationDate | If not available, set null. |
| .details[].memberClaims[].rxNumber | .prescriptionNumber | Direct copy. |
| .details[].memberClaims[].drug.ndcId | .NDC11 | Direct copy. |

**Detail selection rule:** The service **fetches claims** and then returns the **single claim** matching the provided claimId (i.e., claimNumber), after required startDate/endDate validation.

**How these two mappings are used (narrative)**

* **Prescription List (POST /prescriptions/search)**  
  Use this to retrieve a member-scoped **list** of prescriptions over a date range. The Claims Service may return multiple claims per uniqueRxId; the list mapping **deduplicates** to one record per prescription by keeping the claim with the **latest fillDate**. Numeric fields (e.g., daysSupply, quantity) are normalized, and basic demographics/relationship fields are passed through. If membershipResourceIds is empty, all visible members’ prescriptions are considered; otherwise, results are restricted to the provided IDs.
* **Prescription Detail (GET /prescriptions?...claimId=…)**  
  Use this to retrieve the **full detail** for a **specific claim/prescription** identified by claimId. It enriches the list view with additional attributes like **dosage form**, **refills left**, **pharmacy/prescriber display names**, **NDC11**, **prescription number**, **estimatedCost** (computed as payAmount + clientPayAmount), and an optional **expiration date** (nullable). This endpoint still requires the same date range validation.