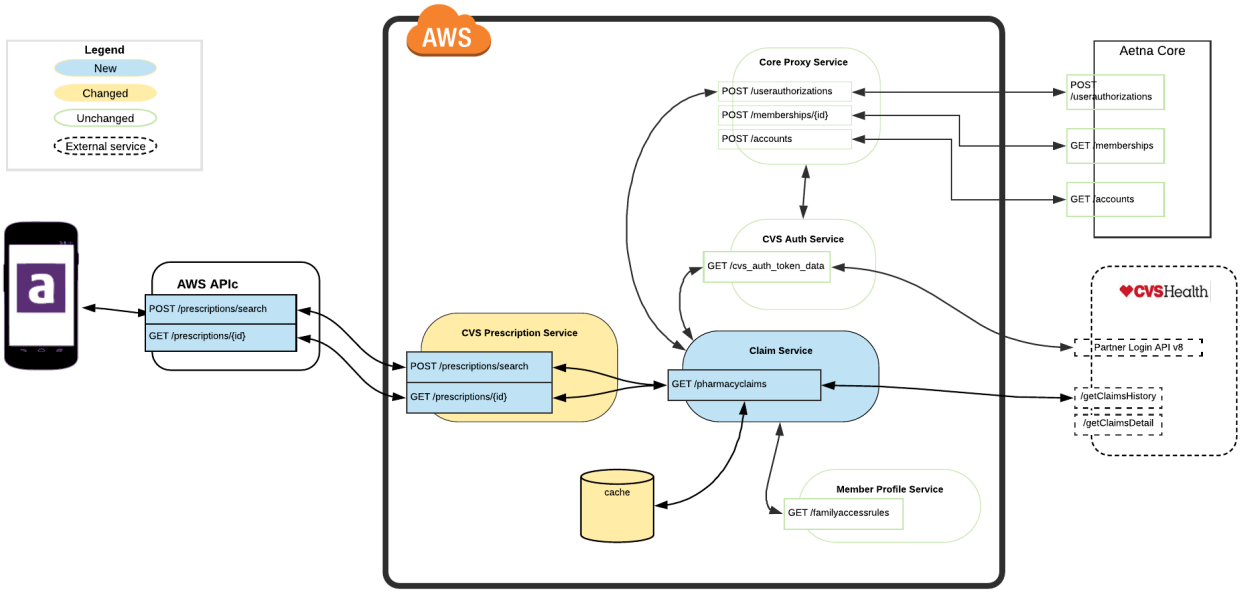
Prescription (Rx) List

Feature

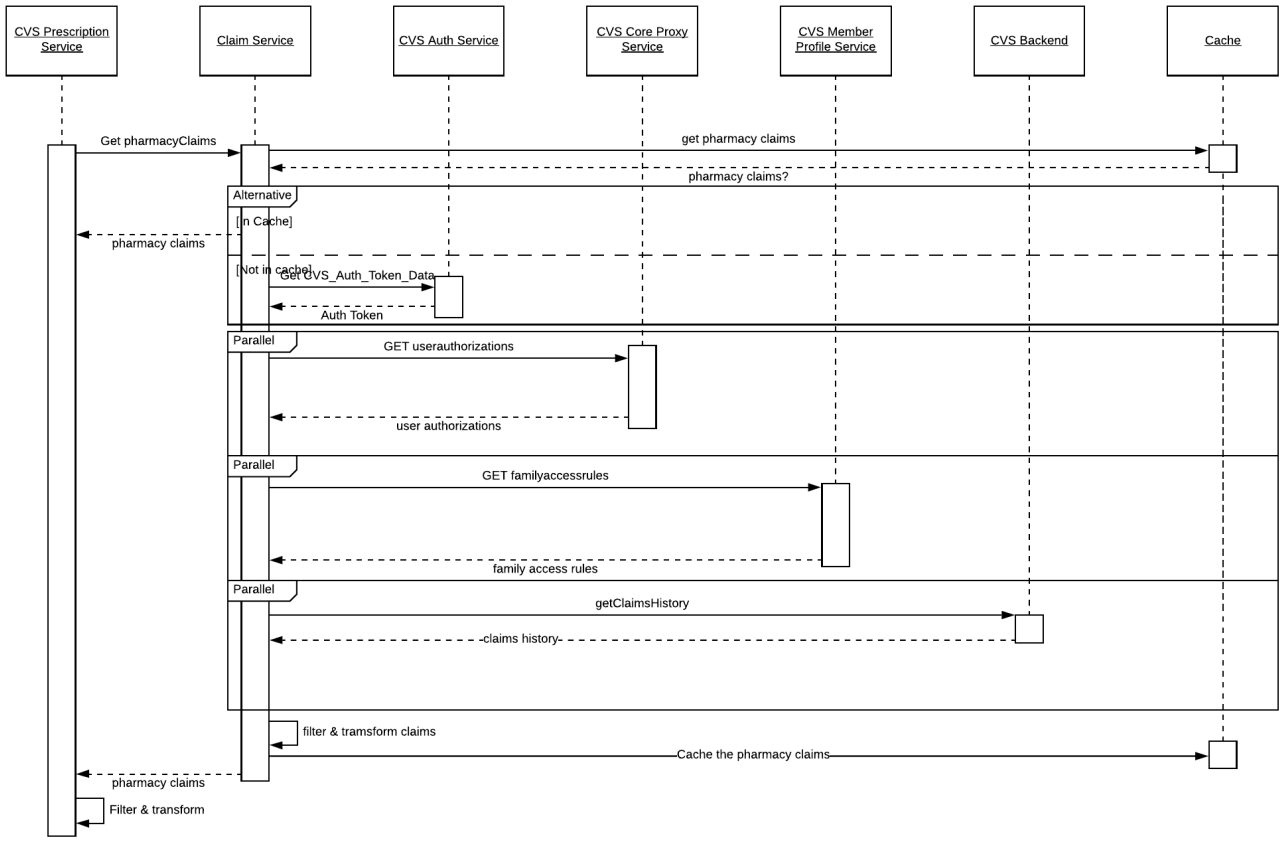
https://rally1.rallydev.com/#/328710528180d/detail/portfolioitem/feature/310857407716?fdp=true

Solution Diagram

Feature: Prescription list



Flow Description



Scenario 1: Client requests for prescription list

- Clients call the POST /prescriptions/search API when requesting the active prescriptions.

- CVS Prescription service will call claims service GET /pharmacyclaims API.
- Claims service will search for the pharmacy claims in the cache. If the claims are cached, return to CVS Prescription service and CVS Prescription service will transform the pharmacy claims object to prescription response and return to the caller. If the claims are not cached, move to next step.
- Claims service will verify that the membership id passed is a valid id and then request for a token from CVS Auth service using /cvs_auth_token_data call.
- Next 3 steps can occur in parallel

1. Claims service will call Core Proxy Service using /userauthorizations to get authorization rules for the requester
2. Claims service will call Member Profile Service using /familyaccessrules to get family access rules for the requester
3. Claims service will call CVS backend using /getClaimsHistory to get claims for the logged in user using the token generated by the cvs_auth_token_data call. The /getClaimsHistory API will return claims for all the members on the plan that are visible to the logged in user .

- Claims Service will apply CVS family access rules to filter the data
- Claims Service will apply UAF rules to filter the data
- Claims Service create the create response object using mapping logics and put into cache
- Claims Service send back the results to CVS Prescription service
- CVS Prescription service will filter and transform the pharmacy claims object to prescription response and return to the caller. @Harrington, Joan K @Behrmann, Ian J Note - when filtering pharmacy claims in this step please use following rules:

1. If in request body the “membershipResourceIds” array is empty, do NO filter, transform all pharmacy claims to prescription response
2. if there are membershipResourceId in the “membershipResourceIds” array of request body, only take pharmacy claims which has its membershipResourceId in the said array to transform to prescription response

Scenario 2: Client requests for prescription detail

- Clients calls the GET /prescriptions/{id} API when requesting the detail of selected prescription
- CVS Prescription service will call claims service GET /pharmacyclaims API. The steps Claims service will execute is same as described above in scenario 1 and return the response to CVS Prescription service.

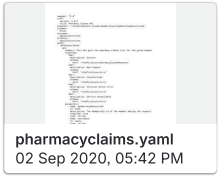
- CVS Prescription Service will filter the response using claimId query parameter and map it based on the transformation below and send back the results to the caller

Swagger Documentation in GitHub

The Swagger artifacts are managed in GitHub for team collaboration. Please use following link to access the latest version of the API documents.

[The Open API document for team to review]

In addition, internal API interface is attached below.


pharmacyclaims.yaml
02 Sep 2020, 05:42 PM

API - Prescription List

POST /prescriptions/search

`POST /v3/aetnahealth_prescriptions/memberships/{membershipResourceId}/prescriptions/search?startDate=2019-04-01&endDate=2019-12-05`

Request object:

```
{
  "membershipResourceIds": [
    "5~263801696+31+1+20180101+788678+C+3"
  ]
}
```

Response:

```
[
  {
    "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"
   }
  ]
 }
]
```

Service Logics

This service will call claims service API GET /pharmacyclaims and use following mapping logic to build response to return to the calling client. If in the claims API response ".details[].memberClaims[]" there are more than 1 claim object that share same value in field "uniqueRxId", select the claim object that has most recent ".fillDate" to include in the /prescriptions/search API response.

If request object contains no membershipResourceId, include prescriptions for all members on the plan that are visible to the logged in user in response. If request object is not empty, return prescriptions in response for the members on the plan whose membershipResourceId are in the request object.

The query parameter "startDate" is required, when it is not passed in the service should throw a bad request error. The query parameter "endDate" is required, when it is not passed in service should throw a bad request error. If the clients don't set the startDate and endDate, at APIC, startDate will be defaulted to 1 year from the current date and endDate will be the current date

Claims API Call Output	Prescriptions API Output	Conversion Logics
.details[].memberClaims[].claimNumber	[].prescriptionList[].claimId	-

.details[].memberClaims[].uniqueRxId	[].prescriptionList[].prescriptionId	-
.details[].memberClaims[].drug.name	[].prescriptionList[].drugName	Title case
.details[].memberClaims[].drug.strength	[].prescriptionList[].drugStrength	String to number
.details[].memberClaims[].daysSupplyQuantity	[].prescriptionList[].daysSupply	String to number
.details[].memberClaims[].dispensedQuantity	[].prescriptionList[].quantity	String to number
.details[].memberFirstName	[].memberFirstName	
.details[].memberLastName	[].memberLastName	
.details[].memberClaims[].fillDate	[].prescriptionList[].lastFilledDate	Collect all the claims that share same value in field “uniqueRxId”, pick the one that has most recent fillDate field and include the object in the response.
details[].memberClaims[].prescriberName	[].prescriptionList[].prescriberName	
details[].memberClaims[].pharmacyName	[].prescriptionList[].fulfilledBy	
.details[].relationshipToSubscriber	[].relationshipToSubscriber	
.details[].membershipResourceId	[].membershipResourceId	
prescriptionsRequest.startDate	[].startDate	
prescriptionsRequest.endDate	[].endDate	

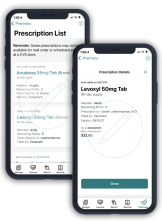
When the service calls claims service GET /pharmacyclaims API, use the request mapping as below:

Caller Parameters	claims API Input	Notes
membershipResourceId path parameter	membershipResourceId path parameter	-
prescriptionsRequest.startDate	startDate query parameter	
prescriptionsRequest.endDate	endDate query parameter	

Error Scenarios

API Response Header - Status Cdde	Error Description	Client Error Response	What to Monitor with New Relic	Requires Alarm Notification
404		200 with empty list	-	-
400		Pass on to caller	-	-
500		Pass on to caller	-	-
	If required query parameter is missing, i.e. startDate or endDate	{ "status": 400, "title": "Bad request", "detail": "The required parameter is missing - " + parameterName }		

Screen Fields Mapping



Screen Fields	API Output	Notes
Auto refill on date label	prescriptionsResponse[].prescriptionList[].lastFilledDate	
Drug label on the mock up screen	prescriptionsResponse[].prescriptionList[].drugName + prescriptionsResponse[].prescriptionList[].drugStrength + prescriptionsResponse[].prescriptionList[].drugForm	The display label on the screen can be created by combining the three API output fields
quantity	prescriptionsResponse[].prescriptionList[].daysSupply	
member	prescriptionsResponse[].memberFirstName	
Remaining refills	prescriptionsResponse[].prescriptionList[].refillsLeft	
Prescribed by	prescriptionsResponse[].prescriptionList[].prescriberName	
Filled by	prescriptionsResponse[].prescriptionList[].fulfilledBy	

API - Prescription Detail

GET /prescriptions

GET `/v3/aetnahealth_prescriptions/memberships/{membershipResourceId}/prescriptions?claimId=200023611694001&startDate=2018-04-12&endDate=2018-12-12`

Response:

{
"claimId": "200023611694001",
"prescriptionId": "674529584",

"lastFilledDate": "2018-04-12",
"memberFirstName": "John",
"memberLastName": "Doe",
"relationshipToSubscriber": "Self",
"membershipResourceId": "5~263801696+31+1+20180101+788678+C+3",
"drugName": "Lipitor",
"drugStrength": "200mg",
"drugForm": "Tablet",
"daysSupply": 5,
"prescriberName": "John Doe",
"fulfilledBy": "Caremark prescription service",
"estimatedCost": 111,
"prescriptionExpirationDate": "2018-04-12",
"prescriptionNumber": "674529584",
"NDC11": "536012297"
}

Service Logics

This service will call GET /pharmacyclaims API to get the pharmacy claim list and go through the list to find the pharmacy claim object with provided claimNumber. Then service will use mapping logic below to create response object to return to caller.

The query parameter “startDate” is required, when it is not passed in the service should throw a bad request error. The query parameter “endDate” is required, when it is not passed in service should throw a bad request error. If the clients don't set the startDate and endDate, at APIC, startDate will be defaulted to 1 year from the current date and endDate will be the current date.

Claims API Call Output	API Output	Conversion Logics
.details[].memberClaims[].claimNumber	.claimId	-
.details[].memberClaims[].uniqueRxId	.prescriptionId	-
.details[].memberClaims[].fillDate	.lastFilledDate	-
.details[].memberFirstName	.memberFirstName	

.details[].memberLastName	.memberLastName	
.details[].relationshipToSubscriber	.relationshipToSubscriber	
.details[].membershipResourceId	.membershipResourceId	
.details[].memberClaims[].drug.name	.drugName	Title case
.details[].memberClaims[].drug.strength	.drugStrength	
.details[].memberClaims[].drug.dosageForm	.drugForm	Title case
.details[].memberClaims[].daysSupplyQuantity	.daysSupply	String to number
.details[].memberClaims[].dispensedQuantity	.quantity	String to number
.details[].memberClaims[].refillsLeft	.refillsLeft	String to number
.details[].memberClaims[].prescriber.fullName	.prescriberName	Title case
.details[].memberClaims[].pharmacyName	.fulfilledBy	Title case
	.estimatedCost	This can be calculated: .memberClaims[].payAmount + .memberClaims[].clientPayAmount
	.prescriptionExpirationDate	If not available, set to null
.details[].memberClaims[].rxNumber	.prescriptionNumber	-
.details[].memberClaims[].drug.ndcId	.NDC11	-

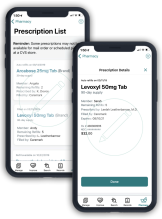
When the service calls claims service GET /pharmacyclaims API, use the request mapping as below:

Caller parameters	Claims API Input	Notes
membershipResourceId path parameter	membershipResourceId path parameter	
startDate query parameter	startDate query parameter	
endDate query parameter	endDate query parameter	

Error Scenarios

API Response Header - Status Cdde	Error Description	Client Error Response	What to Monitor with New Relic	Requires Alarm Notification
400		Pass on to caller	-	-
404		Pass on to caller	-	-
500		Pass on to caller	-	-
	If required query parameter is missing, i.e. claimId, startDate or endDate	{ "status": 400, "title": "Bad request", "detail": "The required parameter is missing - " + parameterName }		

Screen Mapping



Screen Fields	API Output	Notes
Auto refill on date label	prescriptionDetailResponse.lastFilledDate	
Drug label on the mock up screen	prescriptionDetailResponse.drugName + prescriptionDetailResponse.drugStrength + prescriptionDetailResponse.drugForm	The display label on the screen can be created by combining the three API output fields
quantity	prescriptionDetailResponse.daysSupply	
member	prescriptionDetailResponse.memberFirstName	
Remaining refills	prescriptionDetailResponse.refillsLeft	
Prescribed by	prescriptionDetailResponse.prescriberName	
Filled by	prescriptionDetailResponse.fulfilledBy	
Expiration date	prescriptionDetailResponse.prescriptionExpirationDate	
RxNumber	prescriptionDetailResponse.prescriptionNumber	
NDC	prescriptionDetailResponse.NDC11	
Estimated cost	prescriptionDetailResponse.estimatedCost	

API - Claims List

CVS Auth Service

GET /cvs_auth_token_data

There will be a new parameter “nameSuffix” added to the response

```
Request GET /cvs_auth_token_data?membershipId={id}

Response :
{
  "tokenId": "eb6c178f37bf4d7c817cfefa8d8b74e6333ba6261ef1447ab81e63d43b74c73c"
   "members":
   [
     {
       "membershipId":"5~263801696+31+1+20180101+788678+C+3",
       "memberId":"263801696",
       "memberResourceId":"41~263801696",
       "indexId": "1",
       "externalId": "W18168765606",
       "firstName": "BOUDREAU",
       "lastName": "OATWRIGHT",
       "nameSuffix":"Jr",
       "dateOfBirth": "1997-03-10",
       "gender":"M",
       "planId":"770081613829074777110001-A",
       "relationshipToSubscriber":"Self"
     },
     {
       "membershipId":"5~263899614+31+1+20180101+788678+C+3",
       "memberId":"263899614",
       "memberResourceId":"41~263899614",
       "indexId": "2",
       "externalId": "W18168765602",
       "firstName": "IRIS",
       "lastName": "CHAUVET",
       "nameSuffix":"",
       "dateOfBirth": "1990-03-10",
       "gender":"M",
       "planId":"770081613829074777110001-A",
       "relationshipToSubscriber":"Son"
     },
     {
       "membershipId":"5~263899634+31+1+20180101+788678+C+3",   (from Memberships)
       "memberId":"263899634",                                  (from Memberships)
       "memberResourceId":"41~263899634",                       (from Memberships)
       "indexId": "3",                                          (from Partner login
response)
       "externalId": "W18168765603",                            (from Partner login
response)
       "firstName": "JANE",                                     (from Memberships)
       "lastName": "DOE",                                       (from Memberships)
       "nameSuffix":"",                                         (from Memberships)
       "dateOfBirth": "1980-12-01",                             (from Memberships)
       "gender":"M",                                            (from Memberships)
```

```
      "planId":"770081613829074777110001-A",                (from Partner login
response)
      "relationshipToSubscriber":"Daughter"                 (from Memberships)
    }
  ]
}
```

Memberships API	/cvs_auth_token_data (updates)	
membershipResponse.person.nameSuffix	nameSuffix	

GET /pharmacyclaims

```
GET /v3/aetnahealth_claims/memberships/{membershipResourceId}/pharmacyclaims?startDate=2018-02-12&endDate=2018-04-09

Response

[
  {
    "dateOfBirth": "1984-02-18",
    "membershipResourceId": "5~263801696+31+1+20180101+788678+C+3",
    "memberId": "263801696",
    "memberFirstName": "John",
    "memberLastName": "Doe",
    "relationshipToSubscriber": "Self",
    "nameSuffix": "Sr",
    "memberClaims": [
      {
        "claimCOBIndicator": "01",
        "claimNumber": "200023611694001",
        "claimSequenceNumber": "999",
        "clientPayAmount": "60.15",
        "compound": false,
        "dawPenaltyAmount": "0.0",
        "dawPriceDiffAmount": "0.0",
        "daysSupplyQuantity": "5",
        "deductibleAmount": "0.0",
        "deliverySystem": "2",
        "dispensedQuantity": "5.0",
        "excessClaimLimitAmount": "0.0",
        "excessMABAmount": "0.0",
        "fillDate": "2020-04-16",
        "fsaEligibleClaim": "true",
        "gender": "1",
        "genericCoinsuranceIndicator": true,
        "healthcareReimbursementAcctAppliedAmount": "0.0",
        "mchoiceIndicator": "false",
        "medcoMail": false,
        "otherPayerIndicator": false,
        "paidByMedicare": "false",
        "payAmount": "0.0",
```

```
        "personCode": "002",
        "pharmacyNCPDP": "3980958",
        "pharmacyNPI": "326029232",
        "pharmacyName": "CAREMARK PRESCRIPTION SRVC WBP",
        "pharmacyOutOfNetwork": false,
        "pharmacyTypeCode": "0",
        "planId": "770092276746079053911005 1BA",
        "prescriberName": "DUSSEAU, JOSEPH PATRICK",
        "refillOriginator": "0",
        "refillsLeft": "0",
        "rxNumber": "674528227",
        "status": "ACCEPTED",
        "totalCopayAmount": "35.00",
        "uniqueRxId": "97514855016745282 27",
        "prescriber": {
          "address": {
            "country": "0"
          },
          "fullName": "ALBERS, GREGG RICHARD",
          "npiNumber": "1033107701"
        },
        "drug": {
          "abbreviateDrugForm": "TABLET",
          "brand": true,
          "dosageForm": "TABLET",
          "genericAvailable": true,
          "gpiCode": "36150040200320",
          "isSensitive": true,
          "labelName": "ZETIA    TAB 10MG",
          "name": "Prozac",
          "ndcId": "66582041474",
          "packageSize": "500.00",
          "preferred": true,
          "strength": "200mg"
        }
      }
    ]
  }
]
```


Service Logics

This service will call CVS API /getClaimHistory and filter the API call response using the family access rules first and then filter using privacy checks, then apply the mapping logics below to return the API response to the caller.

Family access rules checks

Loop though all the pharmacy claims from getClaimsHistory response :

For each pharmacy claim in the getClaimsHistory response :

- if the pharmacyClaim.indexId matches the logged in member's indexId(use the membershipResourceId in the path, map it to the indexId from the partner login data), skip the family access rules check.

- if the pharmacyClaim.indexId is of a member whose age is less than 18 years(use the date of birth from GET /familyaccesrules response), skip the family access rules check.
- if the pharmacyClaim.indexId does not match the logged in member's indexId and if the member's age >=18 years, check for the corresponding indexId in the familyaccessrules array.
  - If there are no familyaccessrules for that indexId, filter those pharmacy claims.
  - else
    - If the drug.sensitive = true and viewMySensitiveMedications is false, for the member associated with the indexId, then filter out the pharmacy claim. (whenever the drug is sensitive and if the logged in member doesn't have access to see a sensitive drug, we just filter it out).
    - else send that pharmacy claim to the client.

Privacy checks

A dependent may create a HIPAA privacy restriction, such that the subscriber should not be permitted to see any of that dependent's specific healthcare related information. We will need to have logic similar to what is done for health claims. Before responding back to the client with the set of prescriptions, we'll need to gather the distinct list of memberIds, and with these memberIds, plus the EIE headers, execute a call to UAF. UAF will return a response indicating if the individual making the request (known from EIE headers) is permitted to view data for each of the members in the request. UAF will check access entitlement, and privacy restrictions.
Values to check in the response from UAF:

completeAccessEntitlement=true means that there are NO privacy restrictions between this user and this membershipResourceId in the array.
completeAccessEntitlement=false - there's at least one restriction, need to go deeper to determine if that restriction is relevant to prescriptions.
Unlikely in this case. If the individual is requesting access to someone who isn't even part of their plan. Since this comes back from CVS with these checks and associations already done on their side - it should never have a false value.
look deeper to see if that restriction is healthClaims or healthClaim with state of eliminate.
privacyRestrictionType=RES
Prescriptions privacy checks should use the same UAF logic/filtering as our claims service since they are so closely related. For each entitledResolvedIdentifier we'll need to remove prescriptions for that membershipResourceId/memberId if:
restriction.name = healthClaims and restriction.state = eliminate

restriction.name = healthClaim and restriction.state = eliminate
Logic to use for UAF restrictions

Loop through an array for all the members that were requested in UAF request

```
{
  if (completeAccessEntitlement == false) {
    if ( (healthClaims has eliminate) || (healthClaim has eliminate) {
      hide the prescriptions/refills for this dependent
    }
  }
}
```

SDK Output	API Output	Conversion Logics
.details[].dateOfBirth	[].dateOfBirth	Direct copy
.details[].indexId	[].membershipResourceId	Map indexId to membershipResourceId using the partner login data
.details[].indexId	[].memberId	Map indexId to memberId using the partner login data
CVS auth token data	[].memberFirstName	
CVS auth token data	[].memberLastName	
CVS auth token data	[].relationshipToSubscriber	
CVS auth token data	[].nameSuffix	Optional field, if not found in CVS auth token data, do not send back in API response
.details[].memberClaims[].claimCOBIndicator	[].memberClaims[].claimCOBIndicator	Direct copy
.details[].memberClaims[].claimNumber	[].memberClaims[].claimNumber	Direct copy
.details[].memberClaims[].claimSequenceNumber	[].memberClaims[].claimSequenceNumber	Direct copy
.details[].memberClaims[].clientPayAmount	[].memberClaims[].clientPayAmount	Direct copy

.details[].memberClaims[].compound	[].memberClaims[].compound	Direct copy
.details[].memberClaims[].dawPenaltyAmount	[].memberClaims[].dawPenaltyAmount	Direct copy
.details[].memberClaims[].dawPriceDiffAmount	[].memberClaims[].dawPriceDiffAmount	Direct copy
.details[].memberClaims[].daysSupplyQuantity	[].memberClaims[].daysSupplyQuantity	Direct copy
.details[].memberClaims[].deductibleAmount	[].memberClaims[].deductibleAmount	Direct copy
.details[].memberClaims[].deliverySystem	[].memberClaims[].deliverySystem	Direct copy
.details[].memberClaims[].dispensedQuantity	[].memberClaims[].dispensedQuantity	Direct copy
.details[].memberClaims[].excessClaimLimitAmount	[].memberClaims[].excessClaimLimitAmount	Direct copy
.details[].memberClaims[].excessMABAmount	[].memberClaims[].excessMABAmount	Direct copy
.details[].memberClaims[].fillDate	[].memberClaims[].fillDate	Direct copy
.details[].memberClaims[].fsaEligibleClaim	[].memberClaims[].fsaEligibleClaim	Direct copy
.details[].memberClaims[].gender	[].memberClaims[].gender	Direct copy
.details[].memberClaims[].genericCoinsuranceIndicator	[].memberClaims[].genericCoinsuranceIndicator	Direct copy
.details[].memberClaims[].healthcareReimbursementAcctAppliedAmount	[].memberClaims[].healthcareReimbursementAcctAppliedAmount	Direct copy

.details[].memberClaims[].mchoiceIndicator	[].memberClaims[].mchoiceIndicator	Direct copy
.details[].memberClaims[].medcoMail	[].memberClaims[].medcoMail	Direct copy
.details[].memberClaims[].otherPayerIndicator	[].memberClaims[].otherPayerIndicator	Direct copy
.details[].memberClaims[].paidByMedicare	[].memberClaims[].paidByMedicare	Direct copy
.details[].memberClaims[].payAmount	[].memberClaims[].payAmount	Direct copy
.details[].memberClaims[].personCode	[].memberClaims[].personCode	Direct copy
.details[].memberClaims[].pharmacyNCPDP	[].memberClaims[].pharmacyNCPDP	Direct copy
.details[].memberClaims[].pharmacyNPI	[].memberClaims[].pharmacyNPI	Direct copy
.details[].memberClaims[].pharmacyName	[].memberClaims[].pharmacyName	Direct copy
.details[].memberClaims[].pharmacyOutOfNetwork	[].memberClaims[].pharmacyOutOfNetwork	Direct copy
.details[].memberClaims[].pharmacyTypeCode	[].memberClaims[].pharmacyTypeCode	Direct copy
.details[].memberClaims[].planId	[].memberClaims[].planId	Direct copy
.details[].memberClaims[].prescriberName	[].memberClaims[].prescriberName	Direct copy
.details[].memberClaims[].refillOriginator	[].memberClaims[].refillOriginator	Direct copy
.details[].memberClaims[].refillsLeft	[].memberClaims[].refillsLeft	Direct copy

.details[].memberClaims[].rxNumber	[].memberClaims[].rxNumber	Direct copy
.details[].memberClaims[].status	[].memberClaims[].status	Direct copy
.details[].memberClaims[].totalCopayAmount	[].memberClaims[].totalCopayAmount	Direct copy
.details[].memberClaims[].uniqueRxId	[].memberClaims[].uniqueRxId	Direct copy
.details[].memberClaims[].prescriber.address	[].memberClaims[].prescriber.address	Direct copy
.details[].memberClaims[].prescriber.address.country	[].memberClaims[].prescriber.address.country	Direct copy
.details[].memberClaims[].prescriber.fullName	[].memberClaims[].prescriber.fullName	Direct copy
.details[].memberClaims[].prescriber.npiNumber	[].memberClaims[].prescriber.npiNumber	Direct copy
.details[].memberClaims[].drug.abbreviateDrugForm	[].memberClaims[].drug.abbreviateDrugForm	Direct copy
.details[].memberClaims[].drug.brand	[].memberClaims[].drug.brand	Direct copy
.details[].memberClaims[].drug.dosageForm	[].memberClaims[].drug.dosageForm	Direct copy
.details[].memberClaims[].drug.genericAvailable	[].memberClaims[].drug.genericAvailable	Direct copy
.details[].memberClaims[].drug.gpiCode	[].memberClaims[].drug.gpiCode	Direct copy
.details[].memberClaims[].drug.isSensitive	[].memberClaims[].drug.isSensitive	Direct copy
.details[].memberClaims[].drug.labelName	[].memberClaims[].drug.labelName	Direct copy

.details[].memberClaims[].drug.name	[].memberClaims[].drug.name	Direct copy
.details[].memberClaims[].drug.genericAvailable	[].memberClaims[].drug.genericAvailable	Direct copy
.details[].memberClaims[].drug.ndcId	[].memberClaims[].drug.ndcId	Direct copy
.details[].memberClaims[].drug.packageSize	[].memberClaims[].drug.packageSize	Direct copy
.details[].memberClaims[].drug.preferred	[].memberClaims[].drug.preferred	Direct copy
.details[].memberClaims[].drug.strength	[].memberClaims[].drug.strength	Direct copy

When the service calls CVS /getClaimsHistory API, use request mapping logics below

Caller Parameters	CVS API Input	Notes
	param.apiKey	The Aetna service key for calling CVS backend
	param.refId	Use global transaction id from EIE header, select the value of field "eieheadertransactionid". Details please refer to Ref ID for CVS API Calls
CVS auth token data	param.tokenId	
startDate query parameter	param.startDate	
endDate query parameter	param.endDate	

Error Scenarios

SDK Header - Status Code	SDK Details - Status Code	Client Error Response	What to Monitor with New Relic	Requires Alarm Notification

2030	No prescription found	{ "status": 404, "title": "Not found", "detail": "CVS" + statusCode + " - " + statusDesc }	-	-
2001	Service Error - One or more underlying services are down. Please contact the System administrator	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes
9999	Failure	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes
5000	Any of below: • Please provide valid start date(mm/dd/yyyy). • Please provide valid end date(mm/dd/yyyy). • Please provide valid start date	{ "status": 400, "title": "Bad request", "detail": "CVS" + statusCode + " - " + statusDesc }	-	-

	format in (YYYY-MM-DD). • Please provide valid end date format in (YYYY-MM-DD).			
5000	External ID not found.	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes
5000	Please provide valid indexId & planId.	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes
5000	Invalid planId/indexId	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes
1701	Invalid Duration for Get Claims (	{ "status": 400, "title": "Bad	-	-

	Duration more than 18 months).	request", "detail": "CVS" + statusCode + " - " + statusDesc }		
any error	-	{ "status": 500, "title": "Internal Server Error", "detail": "CVS" + statusCode + " - " + statusDesc }	Increment counter metric for "CVS " + statusCode + " - " + statusDesc	yes

Caching

The pharmacy claims data in cache is owned by claims service.

Cache Key

The cache key for pharmacy claims data will be **claims-v1+ "-" + "accountIdentifier.idSource~accountIdentifier.idValue" + “-” + “membershipResourceId" + “-“ + “startDate” + “-” + “endDate”.**

For example (claims-v1+ "-" + "60~QASP1-SUB-181687666" + “-” + “ `5~263801696+31+1+20180101+788678+C+3` **“ + “-“ + “2018-04-05“ + “-” + “2018-12-15“)**

- Note : Need to add the impersonated account ldap **(impersonatingAccountIdentifier.idSource~impersonatingAccountIdentifier.idValue)** too to the key if impersonation is in place.
- Impersonation EIE headers will be as follows "impersonatingAccountIdentifier": { "type": "object", "properties": { "idSource": { "type": "string" }, "idValue": { "type": "string" }, "idType": { "type": "string" } } }

Reading Cache

When claims service read data from cache, it uses the “membershipResourceId“ query parameter, startDate, endDate, accountIdentifier.idSource and accountIdentifier.idValue to construct cache key.

Writing Cache

When claims service write data into cache, it uses query parameter “membershipResourceId”, startDate, endDate, accountIdentifier.idSource and accountIdentifier.idValue to construct cache key, then use this key put the object into cache.