

Returns SDK: ERP integration via Returns API

Content:

- [Overview](#)
- [Authentication flow](#)
- [Use Cases](#)
 - [Claim management](#)
 - [Create a claim](#)
 - [Get claims](#)
 - [Edit an own claim](#)
 - [Edit a received claim](#)
 - [Update claim status](#)
 - [Decide a claim](#)
 - [Decide a claim by sender](#)
 - [Cancel claim](#)
 - [File attachments](#)
 - [Add an attachment](#)
 - [Edit an attachment](#)
 - [Get an attachment](#)
 - [Remove an attachment](#)
 - [Re-open claim](#)
 - [Request further information](#)
 - [Update or submit re-opened claim](#)
 - [Cancel request for information](#)
 - [Return of goods](#)
 - [Request parts](#)
 - [Print shipping label](#)
 - [Dispatch parts](#)
 - [Receive parts](#)
 - [Chat messages](#)
 - [Create a chat message](#)
 - [Get chat messages](#)
 - [Delete a chat message](#)
 - [Dispute claim](#)
 - [Pick-up process](#)

Document version: 1.2, 01/2025

Overview

This guide explains how to use the API methods for the main use cases in Returns from the perspective of an external system which is calling the Returns API. External systems can be webshops, catalogue systems, CRM or ERP systems for example. Therefore Returns acts like a

remote controlled platform which can be fed with data and also from which data can be retrieved.

In principle, the external system sends a **request** to the Returns API, which in turn gives a synchronous **response** back to the calling system.

The Returns API supports the same use cases that can be fulfilled via the Returns website.

Returns API Integration SDK is a zip file that contains a complete and extensive Postman collection to help you figure out and try all possible scenarios and use cases that our public Api offers to better fit your needs.

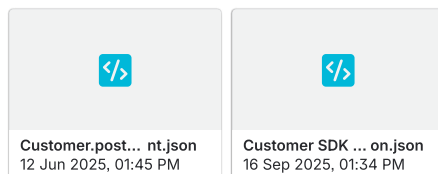
As a prerequisite to use the Returns API, the requesting systems needs to be authenticated.

On top of the SDK content we also suggest you to check our [Swagger UI](#) in which you can find our API definition, requests examples and possible error code returned in each endpoint.

For a complete list of all possible error codes you can also check our [TecCom portal wiki](#).

Prerequisites

Postman **collections** and **environment variables**



In order to make use of the SDK content you need to create an account in [Postman](#) and either use the client application or the browser version. Refer to this [guide](#) on how to import **collections** and the **environment**.

We also suggest you read the Postman official documentation if you are not familiar with the application as it may have cool features to help you during the implementation process and therefore you can get the most out of the collection.

Environment configuration

After you create an account and imported environment and collection you should edit the environment variable imported with the credential and the teccom ids that TecAlliance gives to you. This will help you if you want to export the request contained in the collection in your preferred development language (see the next section for more information).

Email, password, and 2 TecComIDs (Sender and Receiver) are the only variables that need to be set. The rest is automatically set up at runtime if you send the collection in Postman.

Customer

Save Fork 0 Share

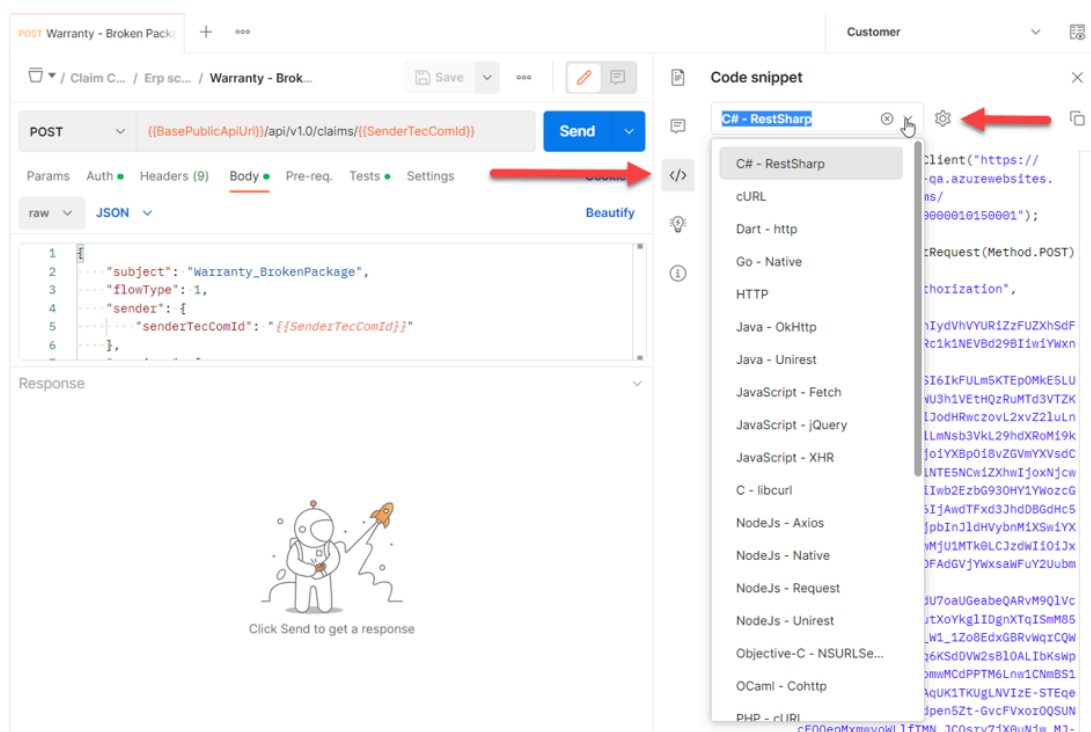
Filter variables

	Variable	Type	Initial value	Current value	...
<input checked="" type="checkbox"/>	BasePublicApiUri	default	https://returnpublicapi-demo.azure...	https://returnpublicapi-demo.azurewebsites.net/	
<input checked="" type="checkbox"/>	email	default	Input email		
<input checked="" type="checkbox"/>	password	default	Input password		
<input checked="" type="checkbox"/>	ReceiverTecComId	default	Input ReceiverTecComId		
<input checked="" type="checkbox"/>	SenderTecComId	default	Input SenderTecComId		
<input checked="" type="checkbox"/>	randomCompanyName	default			
<input checked="" type="checkbox"/>	remark	default			
<input checked="" type="checkbox"/>	access_token	default			
<input checked="" type="checkbox"/>	IndividualResponsesEx	default	[{"individualQuestionId":27,"respons...	[{"individualQuestionId":27,"response":{"questionType":...	
<input checked="" type="checkbox"/>	currentClaimId	default			
<input checked="" type="checkbox"/>	partId	default			
<input checked="" type="checkbox"/>	costId	default			
<input checked="" type="checkbox"/>	performer	default			
<input checked="" type="checkbox"/>	partId2	default			
<input checked="" type="checkbox"/>	currentAttachmentId	default			
	Add new variable				

Disclaimer

Although we provide a 3rd party dependent collection, you don't need to stick with Postman to integrate your API. But Postman offers out of the box a really nice feature that give you the possibility to export the requests you will find in the collection in the language format you prefer or you currently use in your solutions. A lot of languages are supported and we hope that you will like it too!

To do that, you just need to select the request you want to export, and click on the "Code snippet" icon on the right side of the screen as shown in the image. You can also click on the gear icon to change some export options.



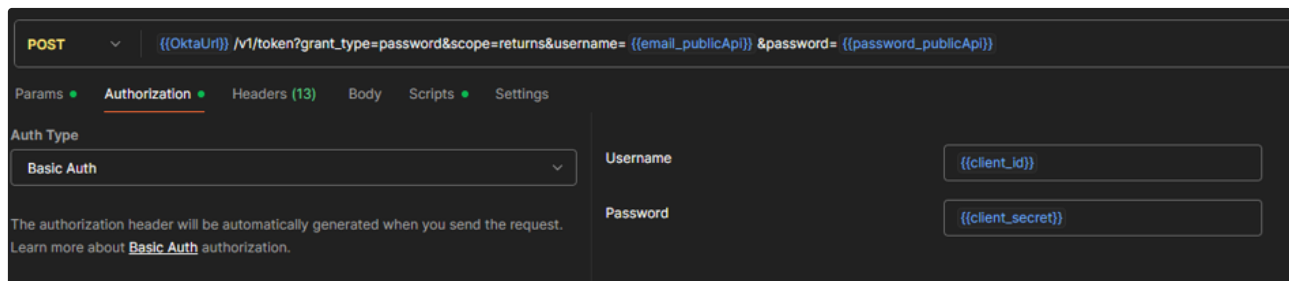
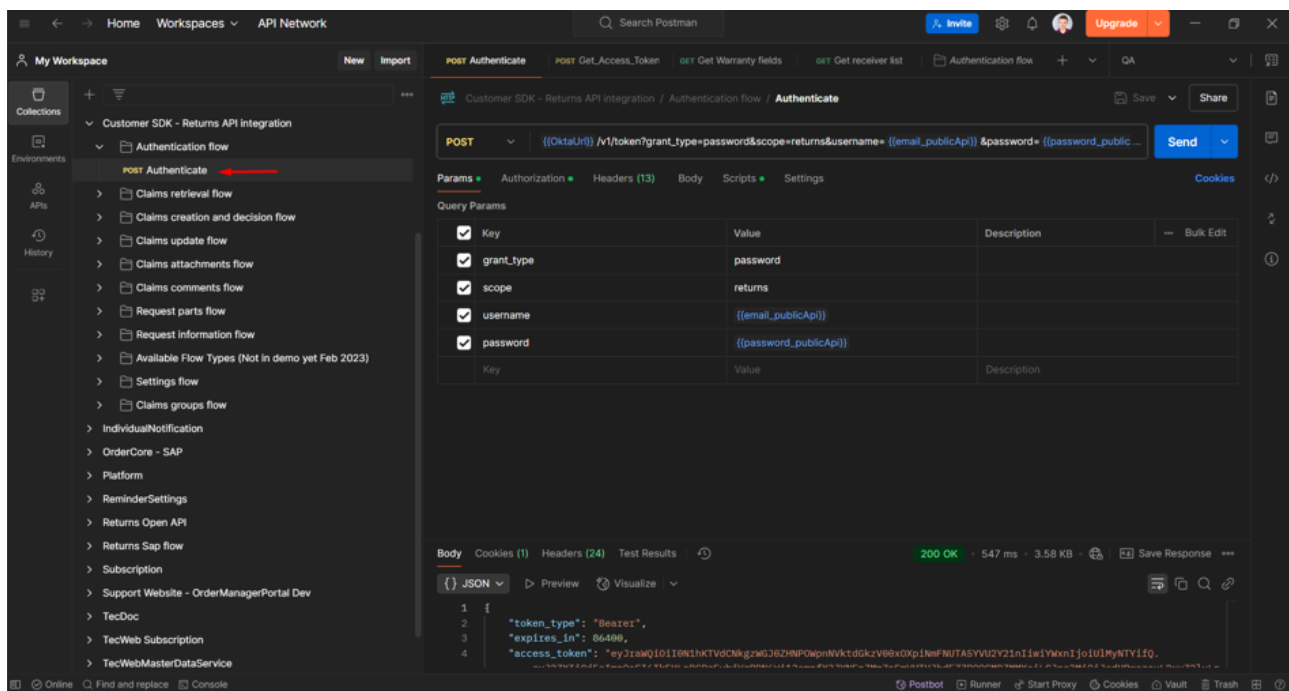
Authentication flow

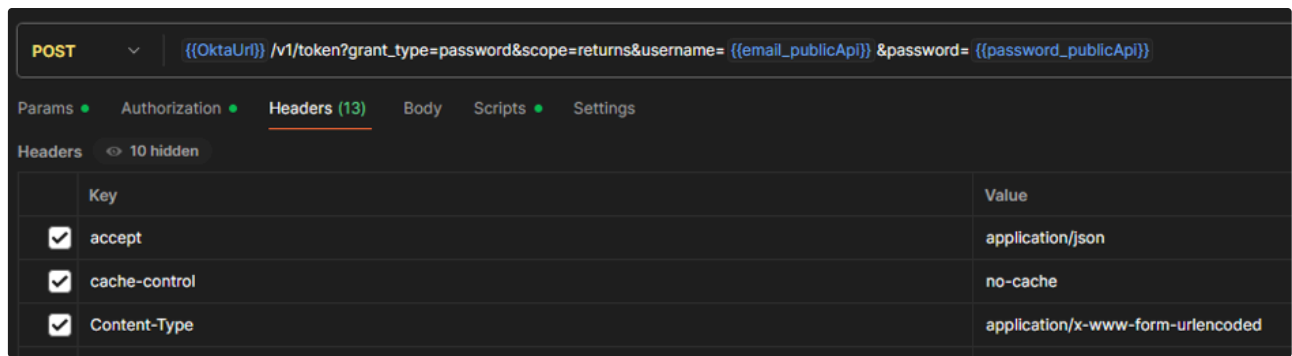
Authentication via API

Authentication is provided by Okta and is JWT based. After you set up your environment you should be able to run the POST Authenticate request contained in the folder.

If everything is set up correctly you will receive a valid json response containing an JWT access token. The received token is automatically saved in your environment and from now on all the request that you will make will use it.

Postman example:





Troubleshooting and Advanced Authentication Details

Two common authentication errors to be aware of:

1. Invalid Password:

Ensure that the password used is correct. You can verify credentials by logging into:

[TecCom](#)

2. Incorrect Authorization Header:

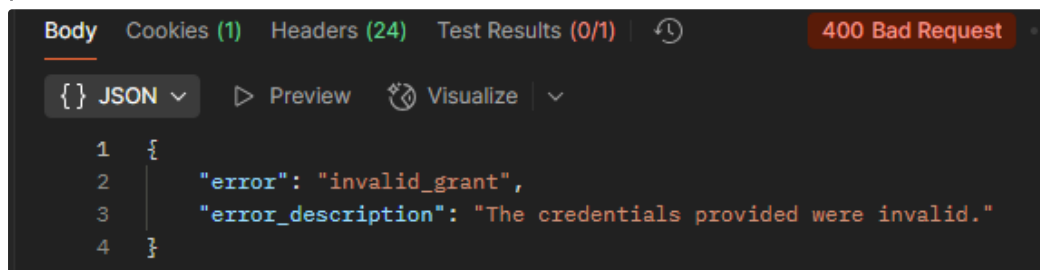
The **Authorization** header must be set to **Basic**, followed by a **base64-encoded** string in the format **client_id:client_secret**.

⚠ Important:

Do not forget the colon (:) separator between the client ID and secret before base64 encoding.

3. Invalid grant. The credentials provided were invalid.

Reserved URL characters (as per RFC 3986) must be URL-encoded when provided as url parameters, i.e. 'Tccom++' → 'Tccom%2B%2B'



Direct Authentication via **curl** (for testing or manual requests)

You may also authenticate manually using a **curl** request to the Okta token endpoint.

Endpoint URL Format:

`https://{yourOktaDomain}/oauth2/default/v1/token`

Required Headers:

```
Accept: application/json
```

```
Cache-Control: no-cache
```

```
Content-Type: application/x-www-form-urlencoded
```

```
Authorization: Basic <base64(client_id:client_secret)>
```

Request Body:

```
grant_type=password username=your_username
```

```
password=your_password
```

Example `curl` Command:

```
curl -X POST
```

```
'https://login.tecalliance.net/oauth2/default/v1/token' \
```

```
--header 'accept: application/json' \
```

```
--header 'cache-control: no-cache' \
```

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

```
\
```

```
--header 'Authorization: Basic
```

```
Y2xpZW50X2lk0mNs4WVudF9zZWNYZXQ=' \
```

```
--data
```

```
'grant_type=password&username=your_username&password=your_password'
```



Note:

Replace `Y2xpZW50X2lk0mNs4WVudF9zZWNYZXQ=` with your actual base64-encoded credentials.

Replace `your_username` and `your_password` with valid user credentials.

To generate the base64 string:

```
echo -n 'client_id:client_secret' | base64
```

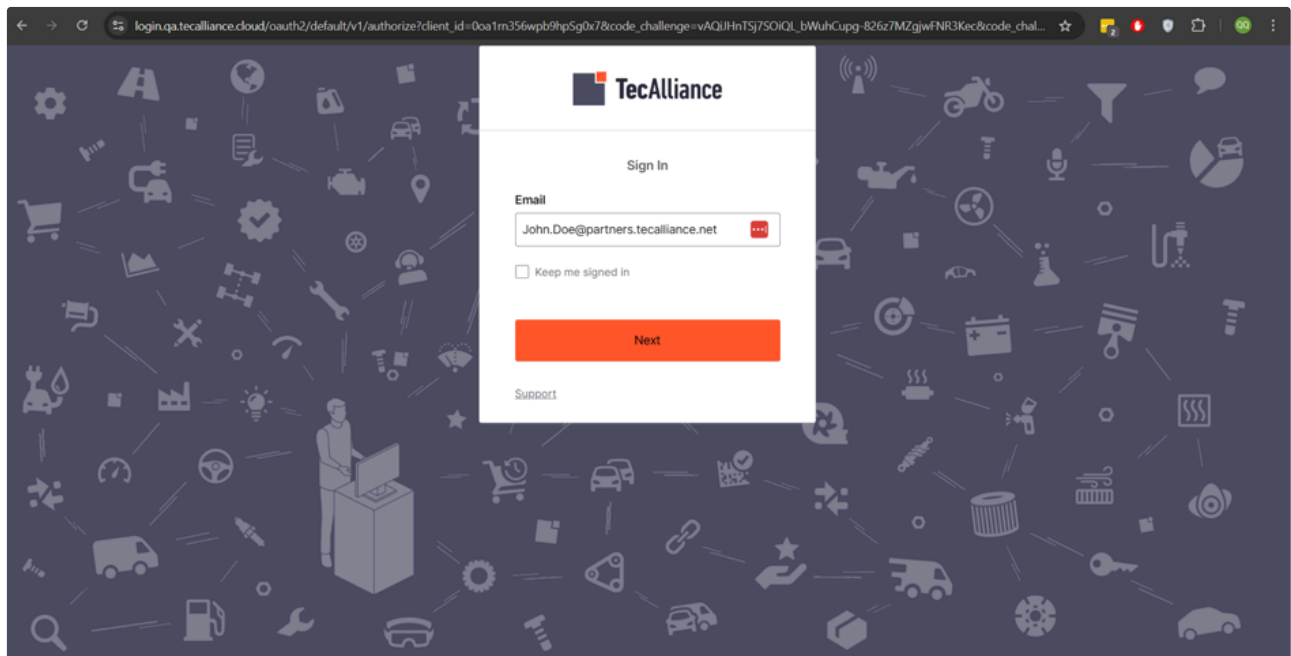
Token Expiry and Reuse Guidelines

- The **JWT access token issued by Okta is valid for 24 hours.**
- Clients **must reuse the same token** for multiple API calls instead of requesting a new token for each request.
- Excessive token requests may lead to future rate limiting and quotas to enforce proper token reuse.

By following these guidelines, you can ensure optimal performance and avoid unnecessary authentication requests.

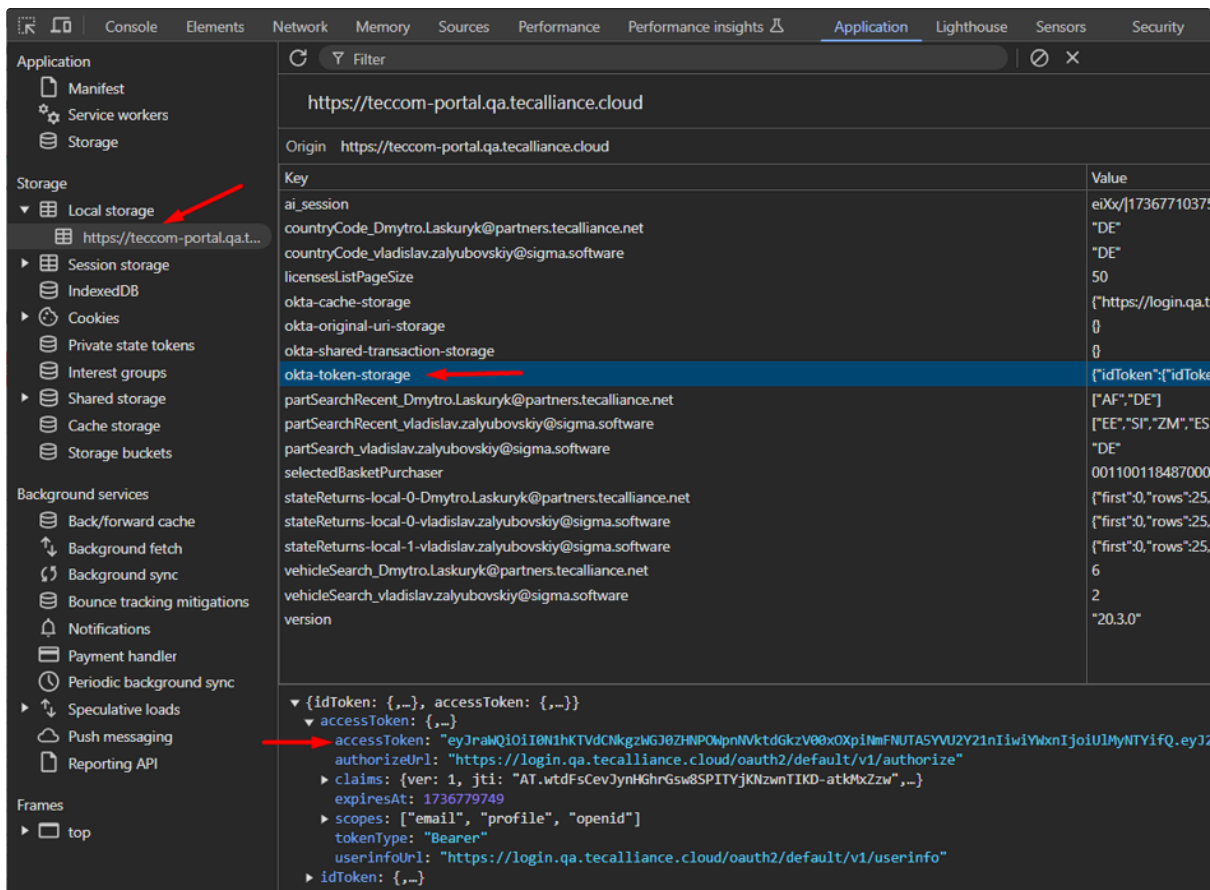
Authentication via Portal

You can authenticate via TecCom Portal and use the access token to perform API requests.



Login page

Once you log in successfully, you can find the **accessToken** in dev tools in local storage at this path:



Path to accessToken

Then copy the value of the **accessToken** and put it in the environment variable in Postman

Customer					Save	Fork	0	Share	...
Filter variables									
	Variable	Type	Initial value	Current value					
<input checked="" type="checkbox"/>	BasePublicApiUrl	default	https://returnspublicapi-demo.azure...	https://returnspublicapi-demo.azurewebsites.net/					
<input checked="" type="checkbox"/>	email	default	qa_returns_aut1@tecalliance.net	qa_returns_aut1@tecalliance.net					
<input checked="" type="checkbox"/>	password	default	teccom-01	teccom-01					
<input checked="" type="checkbox"/>	ReceiverTecComId	default	001000108583000000000010160001	001000108583000000000010160001					
<input checked="" type="checkbox"/>	SenderTecComId	default	001000108583000000000010150001	001000108583000000000010150001					
<input checked="" type="checkbox"/>	randomCompanyName	default							
<input checked="" type="checkbox"/>	remark	default							
<input checked="" type="checkbox"/>	access_token	default	Input access token						
<input checked="" type="checkbox"/>	individualResponsesEx	default	[{"individualQuestionId":27,"respons...	[{"individualQuestionId":27,"response":{"questionType":"...					
<input checked="" type="checkbox"/>	currentClaimId	default							
<input checked="" type="checkbox"/>	partId	default							
<input checked="" type="checkbox"/>	costId	default							
<input checked="" type="checkbox"/>	performer	default							
<input checked="" type="checkbox"/>	partId2	default							
<input checked="" type="checkbox"/>	currentAttachmentId	default							
<input checked="" type="checkbox"/>	OktaUrl	default	https://login.qa.tecalliance.cloud/oa...	https://login.qa.tecalliance.cloud/oauth2/default					

Postman environment variables

Use Cases

The Returns API supports the same use cases that can be fulfilled via the Returns website.

This guide explains how to use the API methods for the main use cases in Returns from the perspective of an external system which is calling the Returns API. External systems can be webshops, catalogue systems, CRM or ERP systems for example. Therefore Returns acts like a remote controlled platform which can be fed with data and also from which data can be retrieved.

In principle, the external system sends a **request** to the Returns API, which in turn gives a synchronous **response** back to the calling system.

As a prerequisite to use the Returns API, the requesting systems needs to be authenticated.

Claim management

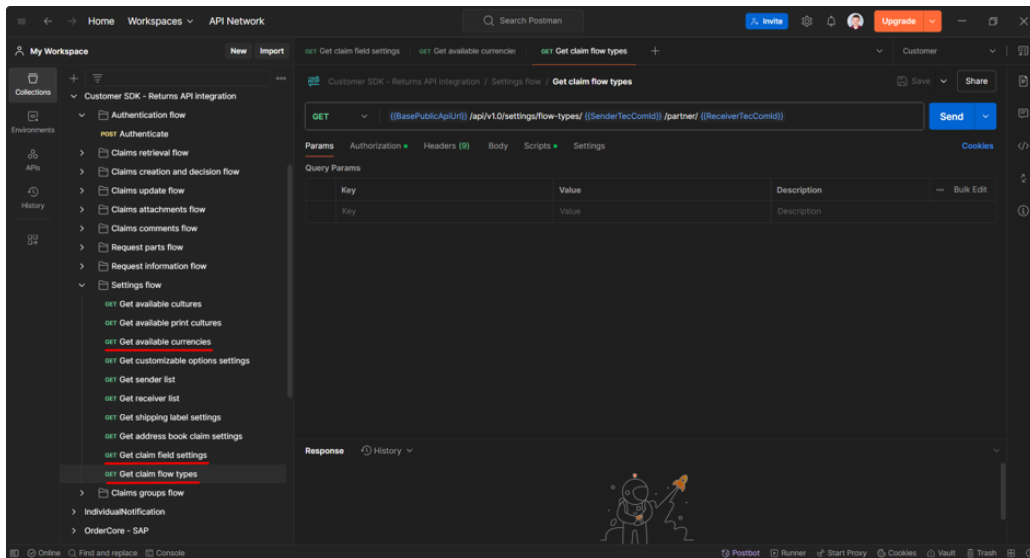
Create a claim

Find request examples in the Postman folder **Claim creation and decision**

Creating a claim via the API in Returns means creating and submitting a claim to the actual recipient in the same time. That means that possible attachments need be added right after claim creation, because attachments are handled by separate methods.

To be able to address a claim to the right claim receiver, a list of possible recipients can be retrieved from Returns, based on the existing TecCom business relations for the sender.

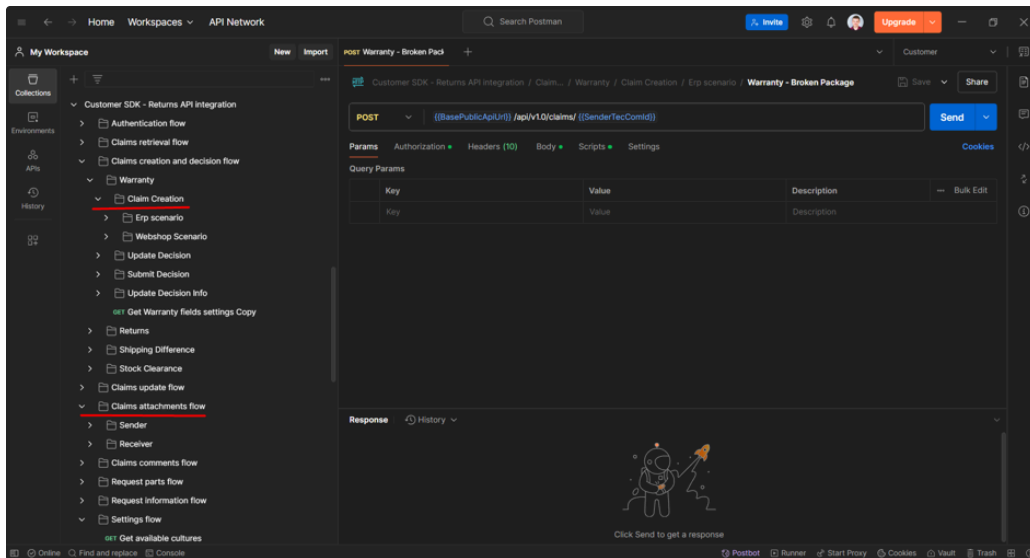
Step	Method	Description
1	GetReceiverList	Retrieves all registered recipients for a specified sender organization.



Setting requests

If all mandatory fields are filled, the claim creation is performed as follows:

Step	Method	Description
5	CreateClaim	<p>Creates a claim in Returns and submits it to the receiver of the claim at the same time.</p> <p>The claim will be in the status "Sent" and then it is not editable anymore.</p> <p>If the receiver has restrictions like a quantity limit of parts per claim, this step will be aborted with an appropriate message.</p>
6	CreateAttachment	<p>It can be used multiple times to upload file attachments for the specified claim.</p> <p>Attachments can be added anytime to a claim, regardless of its status.</p> <p>Once an attachment is added, it can no longer be edited or deleted.</p> <p>If the receiver has restrictions like a file size limit, this step will be aborted with an appropriate message.</p>



Create claims and Create attachments scenarios

Get claims

Find request examples in the Postman folder **Claim retrieval**

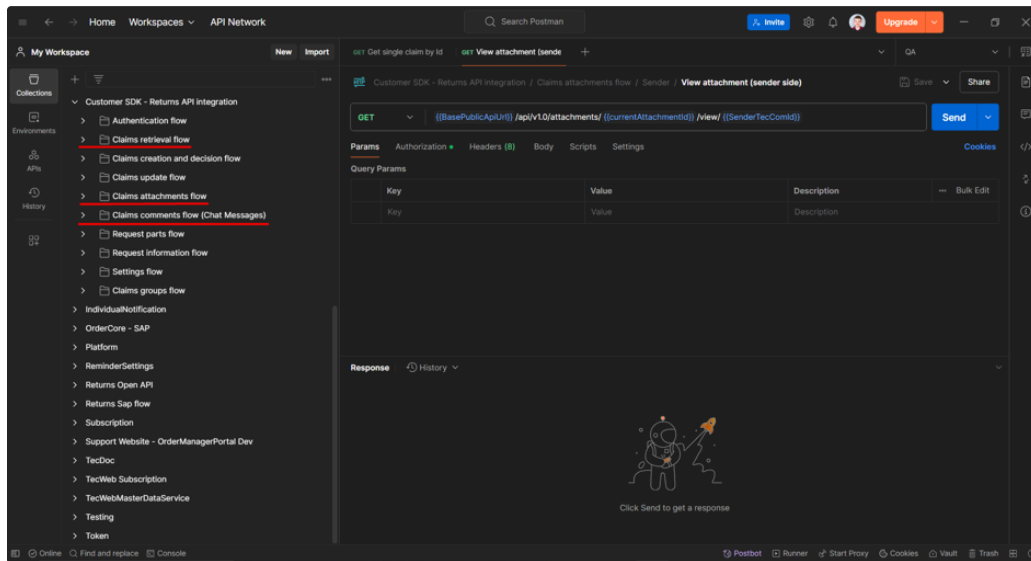
The Returns API supports retrieving data of a single claim as well as a selected list of claims. It's possible to get the entire data of the selected claims as well as limit the result to specified segments of interest.

When reading claims, especially for initial data import into an external system, it makes sense also to grab possible file attachments and comments inside the claim.

Reading claims does not apply any status or data changes to the claims.

Step	Method	Description
1	GetClaims	Retrieves claims via OData expressions for filtering, sorting, expanding etc. Unless filtered out, the claim data contains identifiers of possibly attached files.
2	DownloadAttachment	Optional step: Call this method to download file attachments by identifier (collected via GetClaims in the step before) If only file previews are needed, the method ViewAttachment can be used alternatively.

3	ChatMessages	Optional step: Retrieve possibly existing chat messages of the selected claims (see use case "Claim comments flow")
---	---------------------	---



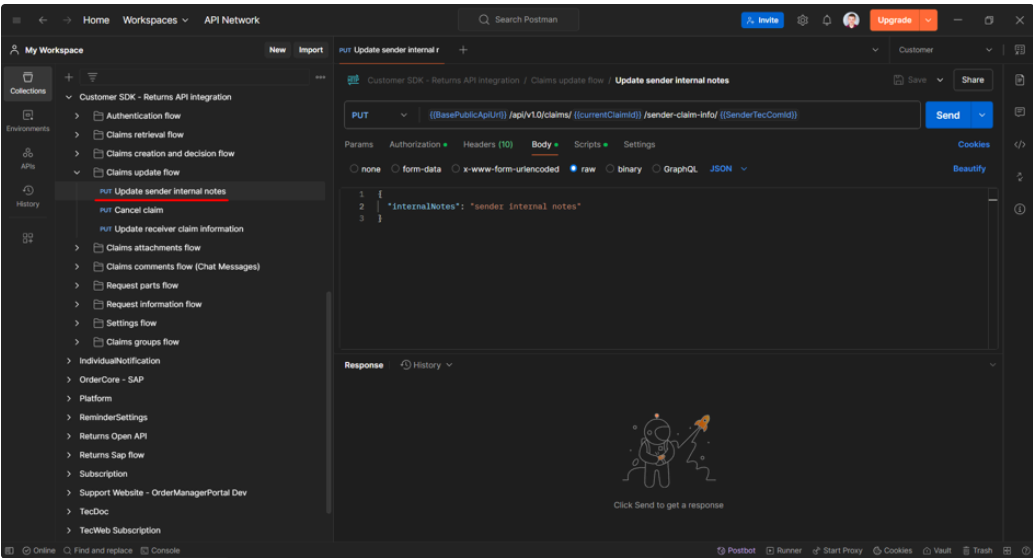
Edit an own claim

Find request examples in the Postman folder **Claims update flow**

For the creator of a claim, there is basically no possibility to edit their own claim after the claim is submitted to the receiver.

The only thing that is possible (besides adding attachments) is to edit the internal comment field, which is not visible to the receiver of the claim.

Method	Description
UpdateSenderClaimInfo	<p>Allows editing the internal comment field for a claim sender.</p> <p>This method can only be called by the original creator of a claim.</p>



Update sender's internal note request

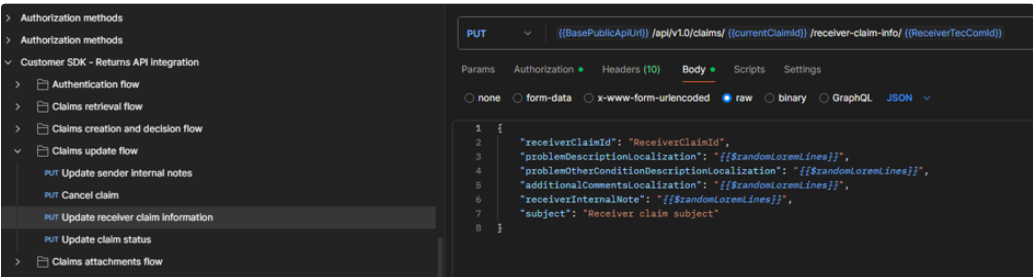
Edit a received claim

Find request examples in the Postman folder **Claims update flow**

For the recipient of a (customer) claim, there are a few options for editing claim data.

Editable fields are mainly comment fields on the receiver side, but also comment fields that are filled by the sender of the claim. The reason is that in multinational scenarios, it's helpful to add localized descriptions to the original sender descriptions.

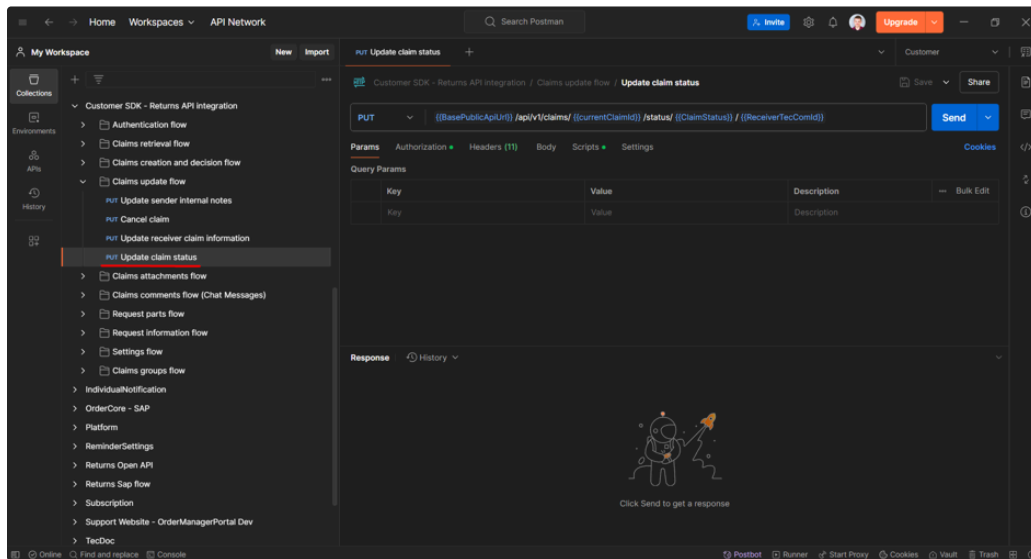
Method	Description
UpdateReceiverClaimInfo	Allows editing of some receiver-side fields as well as sender-side comment fields. This method can only be called by the recipient of a claim.



Update claim status

Maintaining the correct claim and part status improves transparency also for the claim counterpart.

	Method	Description
	UpdateClaimStatus	Sets the claim status to a specific value. This method can only be used by the claim receiver.



Decide a claim

Find request examples in the Postman folder **Claim creation and decision**

Setting the decision of a claim in Returns can be performed in 2 steps: Working on the decision item by item and finally submitting the decision to the claim sender.

The first step in this process is not mandatory.

Claims can only be decided by the receiver of the claim.

Step	Method	Description
1	UpdateDecision	Updates decision related data of a claim without submitting it to the claim sender.

		The claim remains open on the receiver side and it can be worked further on the decision.
2	SubmitDecision	Updates and also submits decision related data of a claim. The claim decision is then visible to the claim sender and the claim is closed.

Decide a claim by sender

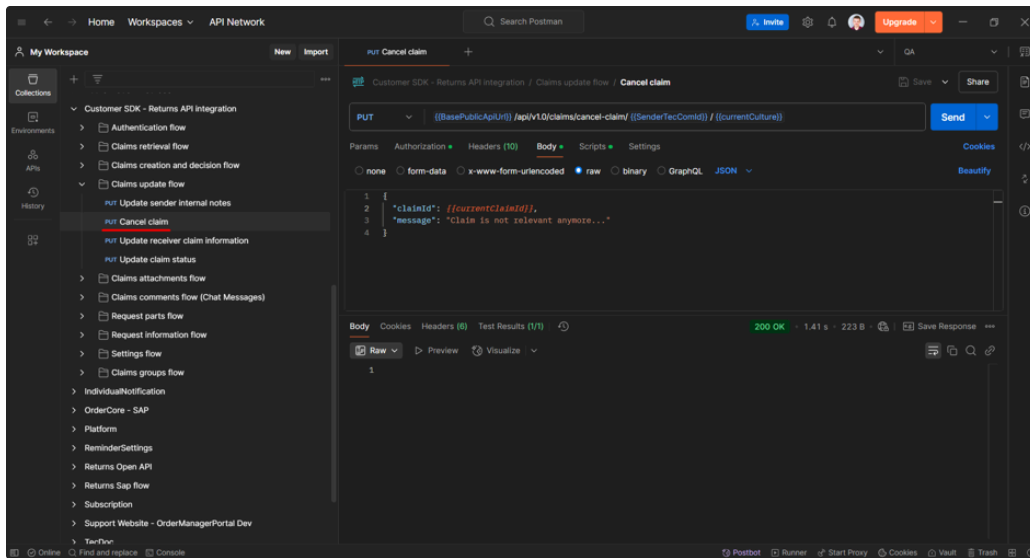
Claims which are not handled by claim receivers can manually be decided by the claim sender.

Step	Method	Description
	SubmitManualDecision	Sets decision related data of a claim by the claim sender. The claim decision is then visible to the claim receiver and the claim is closed.

Cancel claim

Cancelles the claim from the claim sender's side, if the claim is obsolete. Claim cancellation is possible only before the claim has been decided by the claim receiver. The claim will still be visible to both parties in the status "canceled".

	Method	Description
	CancelClaim	Cancels the claim by the claim sender.



Cancel claim request

File attachments

Add an attachment

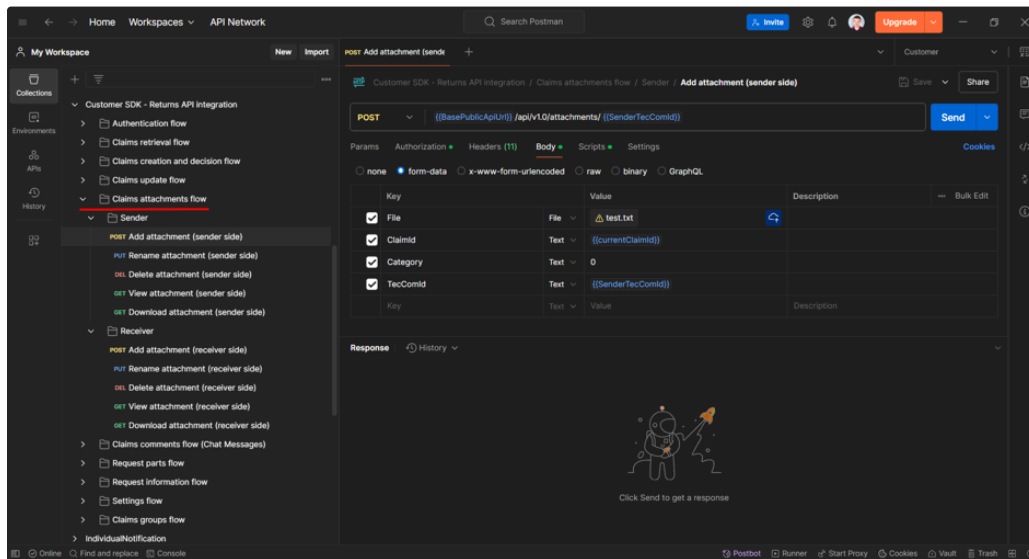
Find request examples in the Postman folder **Claim attachments**

Attachments like photos, videos, or documents can be added anytime during the claim lifecycle from both counterparts of the claim.

Sender attachments are separated from the receiver attachments but can be viewed by both counterparts.

	Method	Description
	CreateAttachment	<p>It can be used multiple times to upload file attachments for the specified claim.</p> <p>The sender cannot add attachments after the claim is sent. However, the receiver can enable the option to allow attachments to be added by the sender at any time through the settings.</p> <p>The receiver can add attachments after the claim is submitted.</p> <p>If the receiver has restrictions like a file size limit, this step will be aborted with an appropriate message.</p>

- To run the request through the Postman collection you need before to specify the physical path of the file you want to upload when running the “Add attachment” request.



Manage attachment

Edit an attachment

Find request examples in the Postman folder **Claim attachments**

Each user can only edit their attachments.

The sender cannot edit their attachments after the claim is submitted.

The receiver can add and edit their attachments after the claim is sent.

	Method	Description
	UpdateAttachment	Allows renaming of an own attachment.

Get an attachment

Find request examples in the Postman folder **Claim attachments**

To access the attachments of a claim, there are different options:

	Method	Description
	ViewAttachment	Shows a preview of the specified attachment without downloading the original file.
	DownloadAttachment	Call this method to download file attachments by identifier If only file previews are needed, method ViewAttachment can be used alternatively.

Remove an attachment

Find request examples in the Postman folder **Claim attachments**

Each user can only delete their attachments.

The sender cannot delete their attachments after the claim is submitted.

The receiver can delete their attachments at any time.

	Method	Description
	DeleteAttachment	Removes an attachment.

Re-open claim

Request further information

Find request examples in the Postman folder **Request information flow**

If the claim receiver asks to have some further information added to the claim by the customer, the claim can be re-opened. The claims sender will then be able to edit the claim and its attachments. This applies only to claims that have already been submitted to the claim receiver and only if the claim has not yet been decided.

	Method	Description
	RequestForInformation	Re-opens a submitted claim by the claim receiver.

Update or submit re-opened claim

Find request examples in the Postman folder **Request information flow**

In the re-opened state, updated information and attachments can be saved to the claim and also finally be re-submitted to the claim receiver. After submission, the claim receiver is able to continue processing the claim, setting the claim back to the state “in progress”.

	Method	Description
	UpdateClaimInformation	Updates a re-opened claim without submitting the update to the claim receiver.
	SubmitRequestForInformation	Re-submits a re-opened claim.

Cancel request for information

Find request examples in the Postman folder **Request information flow**

Cancels the request for information from the claim receiver’s side to be able to continue processing the claim, without needing to wait for the customer’s claim update. The claim will be back in status “in progress”.

	Method	Description
	CancelRequestForInformation	Cancels the re-opening of a claim from the receiver’s side.

Return of goods

Request parts

Find request examples in the Postman folder **Request parts flow**

Actively requesting the parts by the claim receiver to get the parts returned from the customer, is the first step in the physical goods flow.

	Method	Description
	RequestPartsShipping	Initiates the goods return from the customer back to the claim receiver. Allow the receiver to request shipping parts or forbid checking at customer side

Print shipping label

Find request examples in the Postman folder **Request parts flow**

After the receiver requests the parts, the sender may use our API to print a shipping label to return the parts to the receiver. Please note that if parts are not requested, it is not possible to print the shipping or grouped shipping label.

	Method	Description
	PrintShippingLabel	Downloads a shipping label as a PDF file with a QR code that the receiver can use to confirm the goods delivery.
	PrintGroupedShippingLabel	Downloads a combined shipping label for multiple claims as a PDF file with a QR code that the receiver can use to confirm the goods delivery. The claims need to be of the same type.

Dispatch parts

Find request examples in the Postman folder **Request parts flow**

When the claim sender dispatches the parts for the goods return, shipping information can be provided to the parts receiver.

	Method	Description
	SendParts	Informs the claim receiver about the goods dispatch.

Receive parts

Find request examples in the Postman folder **Request parts flow**

When the returned goods arrive at the claim receiver, information about the reception can be entered into the claim.

	Method	Description
	ReceiveParts	Sets information about the received parts.

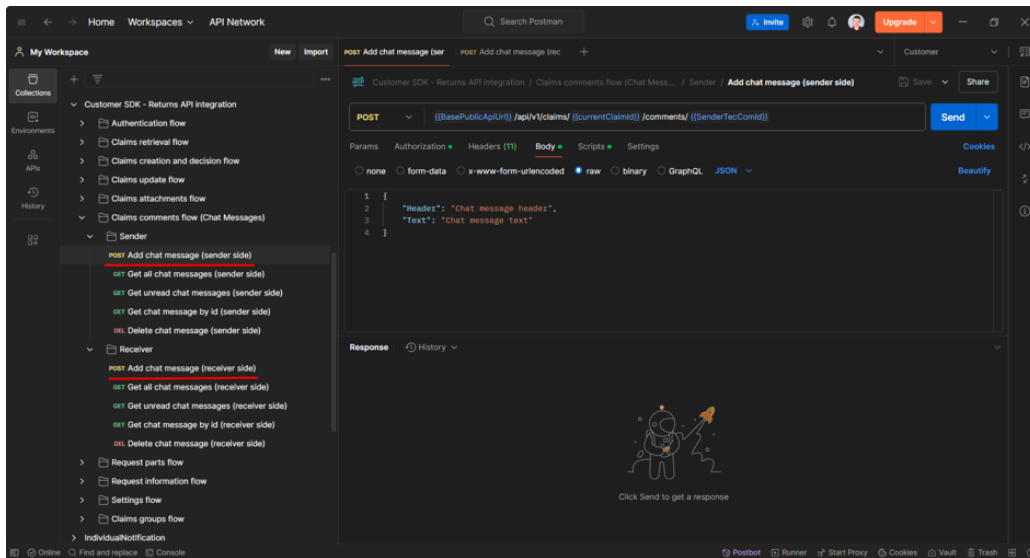
Chat messages

Create a chat message

Find request examples in the Postman folder **Claim comments (chat messages)**

When using the Returns chat for communicating with the claim counterpart, chat messages can be created via Returns API.

	Method	Description
	CreateComment	<p>Creates a new chat message inside the specified claim and posts it directly in the chat.</p> <p>This method can be used both by the sender and by the receiver.</p>



Create a chat message request

Get chat messages

Find request examples in the Postman folder **Claim comments (chat messages)**

For retrieving chat messages of a specific claim in Returns, there are different methods to choose from.

All methods can be used by the claim sender and by the receiver.

	Method	Description
	GetAllComments	Retrieves all existing chat messages of a claim created by the claim counterpart.
	GetUnreadComments	Retrieves only the last unread chat messages from the claim counterpart.
	GetCommentByID	Retrieves only a single chat message, identified by ID.

Delete a chat message

Find request examples in the Postman folder **Claim comments (chat messages)**

It is possible to remove own entries from the chat in Returns, even if the claim counterpart has already replied to it.

	Method	Description
	DeleteComment	Removes a single chat message specified by ID. This method can be used both by the claim sender and by the receiver.

Dispute claim

Find request examples in the Postman folder **Dispute Claim flow**

The Dispute feature lets the Sender to dispute a claim decision in case it was declined or partially accepted.

Key points:

- Sender can dispute a claim only within a certain number of days after the decision.
- Each time Sender disputes again, the dispute level increases until the maximum level is reached.
- When Sender disputes a claim then the receiver can see comment, time of dispute, and any attachments that were provided.
- The system automatically generates a chat message and may also send notifications (email or system messages).

How to use Dispute?

- A claim is decided (Declined or Partially accepted).
- If you disagree with the decision, you can create a dispute.
- You provide a reason and (optionally) upload attachments.
- The receiver is notified and reviews your dispute.
- The receiver can **Accept** or **Reject** the dispute with a comment (optionally).
- If needed, you can dispute again (until the maximum level is reached).

Step	Method	Description
1	DisputeClaim	Creates a new dispute for an existing claim. This allows a claim sender to dispute the decision. A chat message will

		be automatically generated to notify the receiver. The claim will be in status "Dispute"
2	ResolveDispute	Resolves an open dispute. This endpoint is used by the claim receiver to either accept or reject the dispute with comments. The claim will be in status "Decided"

Pick-up process

The idea of the feature is that the claim receiver can decide if he offers pickup service. Pickup is initiated from his side only. Pickup can also happen at workshop location. As a receiver you can initiate a pick-up process for claims with a status **sent**. This feature is about enabling claim receivers, which are using an external system, to offer customers pick-up service for collecting the goods to be returned.

You can use this endpoint to initiate pick-up process



The payload example:

PUT /api/v2/claim/parts/pick-up/{ClaimId}/{TecComId}

```
{
  "transporterName": "transName",
  "trackingId": "TrackNo",
  "trackingLink": "http://tracking-link-example"
}
```

After that **partStatus** will be **PickupInitiated** (value = 7).

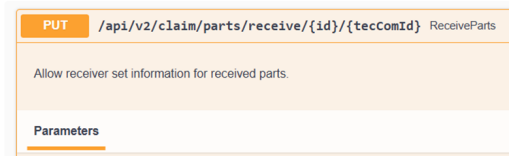


You can check partStatus using this OData query:

/api/odata/v2/claims?TecComId={{TecComId}}&\$expand=partShipping&\$filter=id eq {{ClaimId}}

```
"partShipping": {  
  "partsStatus": 7,  
  "partsReceivedDate": null,  
  "partsSentDate": null,  
  "partsRequestedDate": null,  
  "partsShippingMethod": null,  
  "partsTrackingId": "TrackNo",  
  "shippingAddress": null,  
  "transporterName": "transName",  
  "trackingLink": "http://track.id"  
}
```

You can receive parts using this endpoint:



The payload example:

PUT /api/v2/claim/parts/receive/{{ClaimId}}/{{TecComId}}

```
{  
  "receivedDate": "2025-11-05",  
  "receivingStatus": 1 -- PartsReceived  
}
```

receivingStatus is enum (do not confuse with **PartsStatus** enum):

```
PartsReceivingStatus v Integer($int32)  
Possible values for PartsReceivingStatus:  
0 - NotSentOrReceived  
1 - PartsReceived  
2 - PartsRejected
```

Once its done then **partsStatus** will be **PartsReceived** (value = 5)

The OData response:

```
"partShipping": {  
  "partsStatus": 5,  
  "partsReceivedDate": "2025-11-05T00:00:00Z",  
  "partsSentDate": null,  
  "partsRequestedDate": null,  
  "partsShippingMethod": null,  
  "partsTrackingId": "TrackNo",  
  "shippingAddress": null,  
  "transporterName": "transName",  
  "trackingLink": "http://track.id"  
}
```