EVOTIX

API Guide

Assure Customer API

User Guide: Update Claims Management data via API

Evotix Ltd. Revision 2.0 (July 2024)

EVOTIX

The Assure Customer API is the means by which Evotix provides integration ‘capability to Assure for customers.

The Customer API is available to customers via the public internet and takes the form of a RESTful API. Using the Customer API you can automate processes such as managing users, org unit structure and exporting data for analysis. Making use of the Customer API requires a level of technical expertise so this is typically something that a company’s IT function would handle.

This guide focusses on how to set up the Customer API to update data in claims management in Assure. A separate guide is available for setting up the Customer API.

EVOTIX

Contents

EVOTIX

Managing Claims Management Records via the Customer API

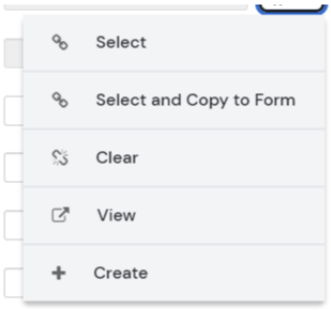
Another of the interactive methods available via the Customer API is the ability to manage Claims Management Records in Assure.

This allows you to create, update and delete records within the Claims Management module.

Limitations

The following lists the current limitations of the API for editing and deleting Claims Management records. It is expected that these will be addressed in future Assure releases:

- The Claims Management reference field cannot be changed via the API. You can only create new records with a unique reference, or update records where there is an existing record for the reference provided. If there are multiple records with the same reference, the API cannot identify which record is trying to be updated and it will error.
- The Claims Management reference field cannot be automatically assigned by the system when creating or updating records, it must be provided in the requests.
- Supporting items cannot be added via the API. This includes actions, attachments, notes, reviews, and links to policies and guidance. If there are default reviewers and approvers set up they will be automatically used on the creation of the record.
- Only the 'Select' option is available in the API when a module reference for either Incident or Vehicle Incident is provided. The API cannot be used with the 'Select and Copy to Form' and 'Create' options as seen in the UI:



- Any default values set within the caption maintenance area are not taken into consideration by the API.

EVOTIX

Pre-configuration for Claims Management Records

Organisational Unit External IDs

Where Organisational Units have been manually created in Assure (i.e. not via the Customer API) there is some pre-configuration required in order to be able then use the Customer API methods. This is required for the API methods to be able to correctly assign Organisational Units to Claims Management records.

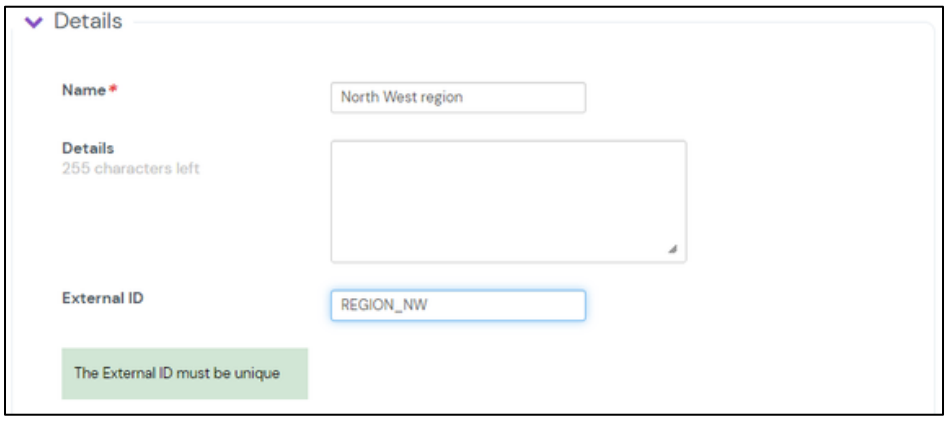
This pre-configuration is only needed for Organisational Units which have been manually added and which do not have an External ID set.

The Assure Organisational Unit hierarchy has a new attribute for each unit called 'External ID'. You will find this in the 'Edit' page of an Organisational Unit and its purpose is to allow an unique external identifier to be associated with each unit. This is necessary because:

1. The existing Organisational Unit names are not unique and therefore cannot be used with an API method to target a specific Organisational Unit.
2. Organisational Unit names can be changed by Assure administrative users so they are not guaranteed to align with the customers IT systems (where user details are being obtained from by the customers integration workflow).
3. Integration workflows should use the immutable unique identifier for Organisational Units so that changes to names (whether in Assure or the source system) do not break the integration workflow. This means Assure needs to be able to configure the unique identifier against each Organisational Unit which is what the External ID field does.

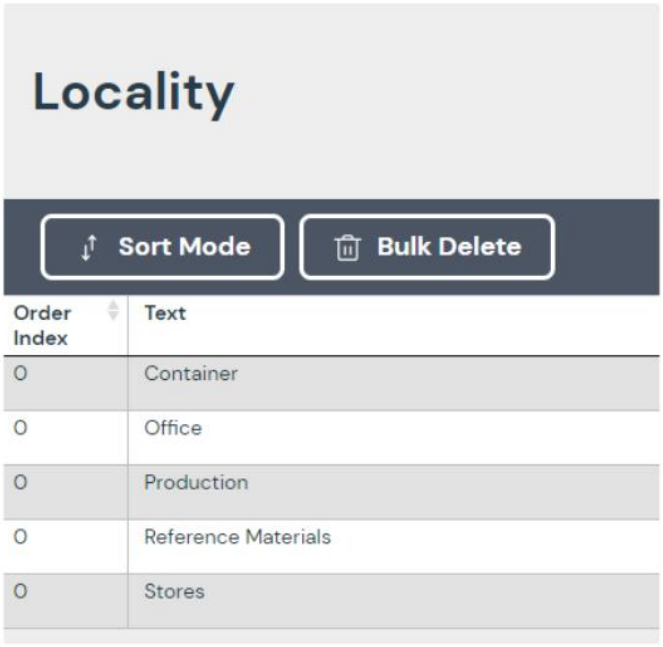
It is strongly recommended that a customer uses their own identifier external ID of an Organisational Unit i.e. the identifier that their IT systems / source data uses for the Organisational Unit (e.g. for a retailer this might be the Shop ID). This removes the need for the customer to maintain a mapping between their Org Unit identifiers and the Assure internal identifier for an Org Unit.

The external IDs for the Organisational Units can be configured manually using the Assure UI (screenshot below shows an example setting the Organisational Unit external ID for the "North West region" to be REGION_NW). This is fine for testing but for ensuring that the Organisational Unit hierarchy is in sync with the customers user management system we have a bulk import/update tool which can be used.

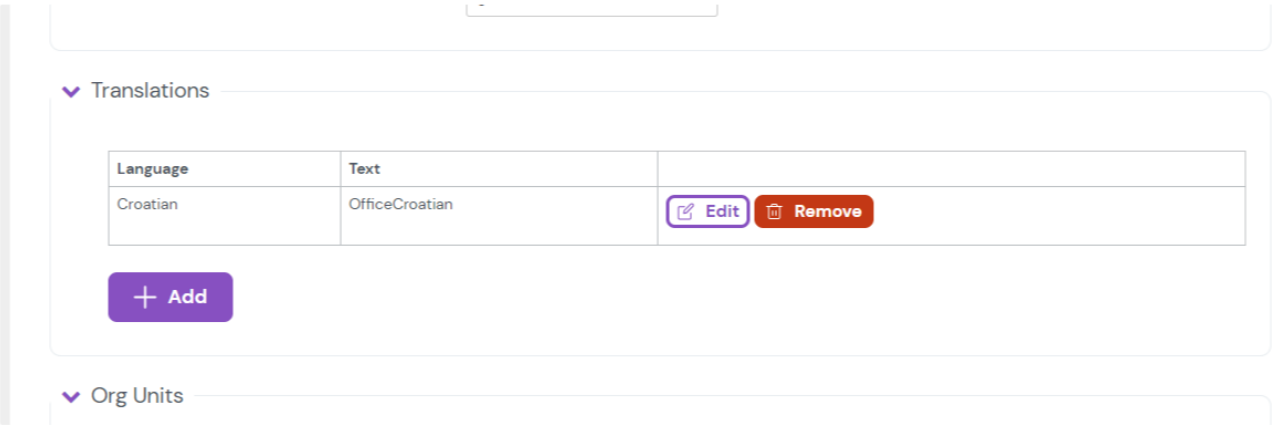


Picklist Values

For any of the fields in a Module record that has values that come from the 'Picklist Data Dictionary' in Assure then the value provided in the API request must match a value in the 'Picklist Data Dictionary' for that field and can be in the customer's default language or base system language (English-UK). For example, the 'Locality' field is a picklist value for Modules. If you locate this value in the Assure UI you will see the values that exist for the field (see example in the screenshot below).

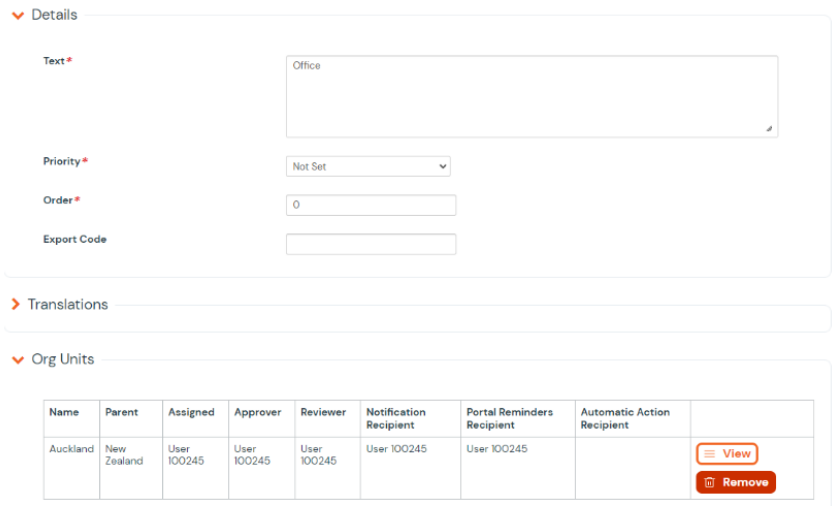


If you have translations for the picklist value and the language is the same as the default customer language this can also be supplied via API. Example, in the screenshot below, if the customer's default language is in Croatian, then the value 'OfficeCroatian' can be supplied via API. Validation is performed in the request to ensure that the 'Locality' field takes only one of these values or no value. This is then the case for all picklist values.



Picklist values can also be restricted by Org-Unit. Using the same example above in the first screenshot the value 'Office' exists for the 'Locality' picklist dictionary value. Here you can restrict 'Office' to only be allowed as a value for a specific org unit. This validation will then work via API as it would in Assure. Meaning that if you then try to create a Module record with 'Office' as the 'Locality' but the org unit supplied in the request is not for the org unit 'Office' was restricted to and so the request would fail. See the second screenshot to see how this restriction can be set up.

The screenshot below shows how this restriction can be set up.



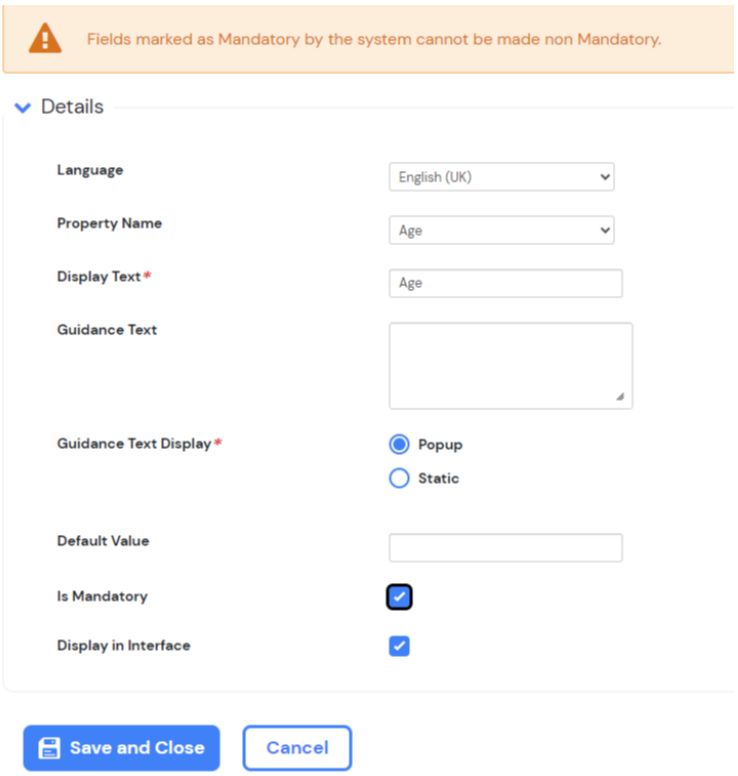
Custom Mandatory Values

In Assure you can set up custom mandatory values for each module, which are then validated on creation and update of records in those modules. This validation is also honoured by the API.

Any field that has been set to mandatory in Claims Management records must be provided in the API request even if the schema does have it as a required field.

These fields are managed via caption maintenance in Assure. The example below shows the set up to make 'Age' a mandatory field. The field must also have the 'Display in Interface' selected for the mandatory validation to occur in the API.

EVOTIX



If a default value is set in caption maintenance, the API does not currently take that into consideration.

EVOTIX

Claims Management JSON object

When creating/updating a Claims Management record the record's details need to be provided in the form of a JSON object. JSON is the most commonly used syntax for describing data objects in RESTful APIs, for an introduction to JSON see this guide. The OpenAPI schema for the Customer API contains the formal definition of the JSON structure i.e. the claimsManagementPOSTRequestobject. The OpenAPI schema is the master definition of the API methods and data objects, it should always be consulted to understand the required fields, field types, max data lengths and string patterns, plus descriptions of the behaviour associated with the use of each field (or its omission). Some software tools and platforms can consume the OpenAPI schema to automate the process of generating the correct JSON and if this is available it should be used. To aid understanding of how the JSON fields in the Claims Management JSON object relate to the resulting Claims Management record setup in Assure the following sections contain some worked examples.

The text encoding to be used for all interactions with the Customer API is UTF-8. This is pretty much the standard today for software tools and platforms however it is important to check that you are using UTF-8 as if not then when you get foreign characters in the data or other symbols like emoticons these will not appear correctly in Assure.

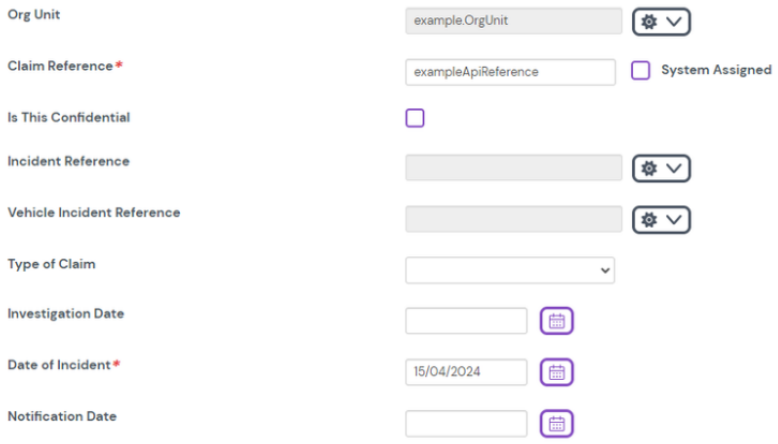
As JSON objects are described using plain text it is possible to hand craft these objects for initial testing. However, it is VERY IMPORTANT to ensure that when generating JSON objects to send to the Customer API you use a proper JSON library or a tool with native JSON support. This is because JSON relies on 'escaping' for certain data values, if this escaping is not done correctly it will lead to API errors and can also be a source of security vulnerabilities.

EVOTIX

Minimum Claims Management data

This example shows the minimum possible Claims Management data which can be used to create or update a Claims Management record,

```
{
    "reference": "exampleApiReference",
    "orgUnitExternalId": "example.OrgUnit",
    "incidentDate": "2024-04-15"
}
```



EVOTIX

Claims Management record with settlement details

This example shows the same Claims Management record with settlement details.

```
{
    "reference": "exampleApiReference",
    "orgUnitExternalId": "example.OrgUnit",
    "incidentDate": "2024-04-15",
     "settlementDetails": {
        "interimPaymentDate": "2024-04-22",
        "settlementDate": "2024-04-22",
        "costs": {
            "estimated": {
                "initialReserve": 1,
                "currentReserve": 2,
                "plaintiffCosts": 3,
                "defenceCosts": 4,
                "vat": 5,
                "excess": 6
            },
            "actual": {
                "interimPayment": 7,
                "plaintiffCosts": 8,
                "defenceCosts": 9,
                "vat": 10,
                "excess": 11
            }
        },
        "settlementComments": "example.settlementComments"
    }
}
```

The Claims Management Record details will be the same as the 'minimum Claims Management data' screen shot above. The difference in this case is that the settlement details are included in the Settlement section.

EVOTIX

EVOTIX

Claims Management record with everything defined

This example shows the same Claims Management record with all the possible fields defined.


```
{
   "reference": "exampleApiReference",
    "orgUnitExternalId": "example.OrgUnit",
    "isConfidential": false,
    "incidentReference": "exampleIncidentReference",
    "vehicleIncidentReference": "exampleVehicleReference",
    "typeOfClaim": "Property",
    "investigationDate": "2024-03-14",
    "incidentDate": "2024-04-15",
    "notificationDate": "2024-04-22",
    "description": "exampleDescription",
    "location": "Car Park",
    "locality": "Gotham",
    "natureOfIncident": "Other",
    "incidentDetails": "exampleIncidentDetails",
    "manager": "exampleManager",
    "probability": "exampleProbability",
    "stageOfClaim": "exampleStageOfClaim",
    "currentPosition": "appeal",
    "outcome": "exampleOutcome",
    "costCentre": "technology",
    "settlementDetails": {
        "interimPaymentDate": "2024-04-22",
        "settlementDate": "2024-04-22",
        "costs": {
            "estimated": {
                "initialReserve": 1,
                "currentReserve": 2,
                "plaintiffCosts": 3,
                "defenceCosts": 4,
                "vat": 5,
                "excess": 6
            },
            "actual": {
                "interimPayment": 7,
                "plaintiffCosts": 8,
                "defenceCosts": 9,
                "vat": 10,
                "excess": 11
            }
        },
        "settlementComments": "examplesettlementComments"
    },
```

EVOTIX


```
    "interestedParties": {
        "insurerReference": "exampleInsurerRef",
        "insurerDetails": "exampleInsurerDetails",
        "brokerDetails": "exampleBrokerDetails",
        "defenceSolicitor": "exampleDefenceSolcitor",
        "lossAdjuster": "exampleLossAdjuster",
        "otherParties": "exampleOtherParties",
        "brokerReference": "exampleBrokerReference"
    },

"claimants": {
        "surname": "Parker",
        "forename": "Peter",
        "address1": "New York street",
        "address2": "Forrest Hill",
        "address3": "Park",
        "town": "Glasgow",
        "county": "Lanarkshire",
        "postcode": "examplePC",
        "dateOfBirth": "2023-05-21",
        "dateOfDeath": "2024-02-11",
        "age": 25,
        "sex": "male",
        "occupation": "other",
        "shift": "early",
        "lostTimeDays": 56,
        "injuryType": "break",
        "apparentCause": "electrocution",
        "partOfBodyAffected": "eye",
        "resultantDisabilityOrHarm": "exampleResultantDisabilityOrHarm",
        "plaintiffName": "exampleplaintiffName",
        "plaintiffSolicitor": "examplePlaintiffSolicitor"
    },
    "otherInformation": {
        "defenceNature": "exampleDefenceNature",
        "namesOfExpertWitnesses": "examplenamesOfExpertWitnesses",
        "negligenceOrCausationAdvice": {
            "isInternal": false,
            "isExternal": false,
            "hasExchangedWitnessReports": true,
            "notes": "exampleNotes"
        },
        "quantumAdvice": {
            "isInternal": false,
            "isExternal": false,
            "hasExchangedWitnessReports": true,
            "notes": "exampleNotes"
        },
        "isComplaint": true,
        "numberOfPeopleInjured": 9,
        "isAlternativeDisputeResolution": true,
        "alternativeResolution": "exampleAlternativeResolution",
        "isStructuringFeasible": true,
        "isAcceptableToPlaintiff": true
    }
}
```


It is not necessary to set all the fields when creating or updating a Claims Management record. Any of the fields which are not required can be omitted in which case the default value for the field will be applied. See the description against each field in the OpenAPI schema for details of the default behaviour.

EVOTIX

Creating a Claims Management Record

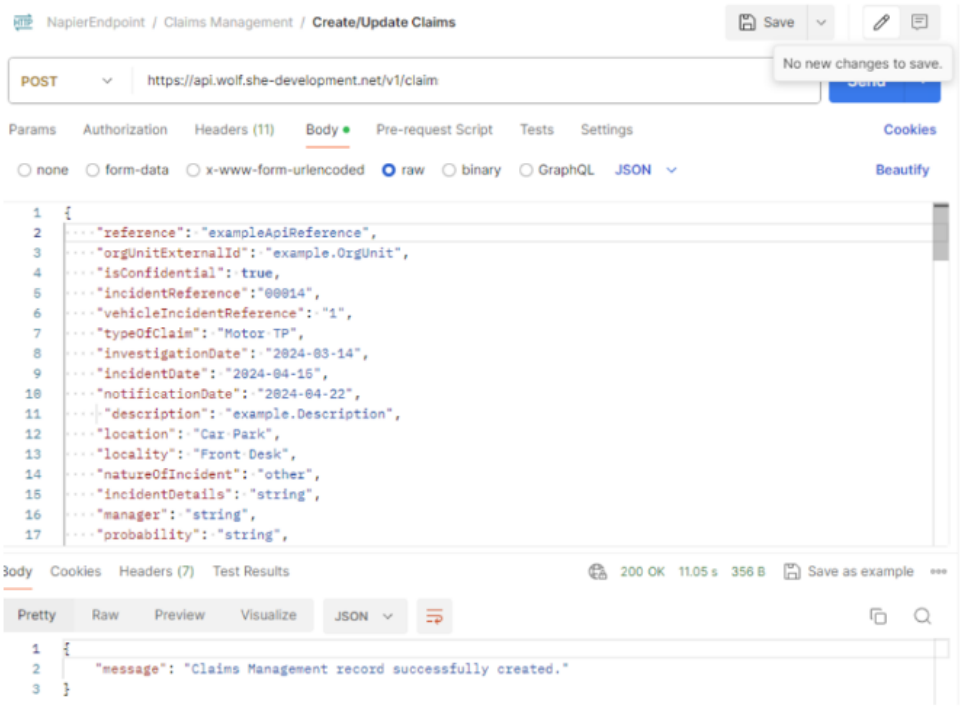
The Customer API allows the creation of Claims Management records in Assure using the /v1/claim API method with the POST verb. The Claims Management record's details are supplied in the body of the API request using the Claims Management JSON object (see section above). Records are uniquely identified by their reference which is the value in the reference field of the Claims Management JSON object. If there are no Claims Management records for the reference provided in the request then the /v1/claim API method will create the record using the details in the Claims Management JSON object.

If there is an existing Claims Management record for the external ID then the /v1/claim API method will update the existing Claims Management record to match the supplied details (see the 'Update a Claims Management' section below for details on the behaviour when updating an existing Claims Management record). Note: If there are multiple Claims Management records with the same reference that is provided in the request this will result in an error response (HTTP error 409).

Before attempting to use the /v1/claim API method make sure you understand the Claims Management JSON object (see the section above) and that you have the required details to access the API (i.e. the URL prefix, API key, etc), follow the Getting Started guide if you don't have these details already. The following table sections show how to create a Claims Management record using the /v1/claim API method in a variety of software tools / platforms:

Any HTTP response code is the range 200-299 should be treated as success (do not use the contents of the response body for identifying success or failure as these may change). Implement effective error handling as per the guide for error handling.

EVOTIX

Postman API platform	The screenshot below shows a successful 'create Claims Management record' request for a customer. The API key is supplied via the 'Authorization' tab (see the Getting Started guide for how to setup the API key). The response section at the bottom shows the result Status: 200 OK which indicates that the record's creation was successful (also displayed is the response message from the Customer API confirming the successful creation). NB: With tools like this where the Claims Management JSON object is manually entered you must ensure the contents of any string values are properly escaped. 

<table>
<tr>
<td>Windows Powershell</td>
<td>The code block below shows the few lines of PowerShell script required to setup the Claims Management Object JSON and to make the ‘create claims management record’ request to the Customer API for a customer. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.


```
$ClaimsManagementObjectJSON = @{
    "reference"             = "exampleApiReference"
    "orgUnitExternalId":    = "example.OrgUnit"
    "incidentDate":         = "2024-04-15"
} | ConvertTo-Json
Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri https://api.wolf.sheassure.net/v1/claim`
    -Method Post `
    -ContentType 'application/json' `
    -Body $ClaimsManagementObjectJSON`
```


The following shows the output from the above PowerShell script code being run where a successful response is generated. The StatusCode: 200 indicates that the create Claims Management request was successful. Some response lines have been removed for brevity.


```
StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Claims Management record successfully created."}
```
</td>
</tr>
<tr>
<td>Python</td>
<td>The code block below shows the few lines of Python code required to setup the Claims Management Object JSON and to make the ‘create claims management’ request to the Customer API for a customer. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.


```
# pip install requests
import requests
claims_management_object = {
    "reference": "exampleApiReference",
    "orgUnitExternalId": "example.OrgUnit",
    "incidentDate": "2024-04-15"
}
headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}
r = requests.post("https://api.wolf.sheassure.net/v1/claim", headers=headers, json =
claims_management_object)
r.raise_for_status()
print(f"StatusCode={r.status_code}")

print(f"Body={r.content}")
```


The following shows the output from the above Python code being run where a successful response is generated. The StatusCode=200 indicates that the claims management record creation was successful.


```
StatusCode=200
Body=b'{"message": "Claims Management record successfully created."}'
```
</td>
</tr>
</table>

EVOTIX

Updating a Claims Management Record

The Customer API allows the updating of claims management records in Assure using the /v1/claim API method with the POST verb. The claims management record details are supplied in the body of the API request using the claims management JSON object (see section above). Claims management records are identified by their reference, which is the value in the reference field of the claims management JSON object. If there is an existing claims management record for the reference provided then the /v1/claim API method will update the existing claims management record to match the supplied details (see the 'Update a claims management record' section below for details on the behaviour when updating an existing claims management record). Note: If there are multiple claims management records with the same reference that is provided in the request this will result in an error response (HTTP error 409).

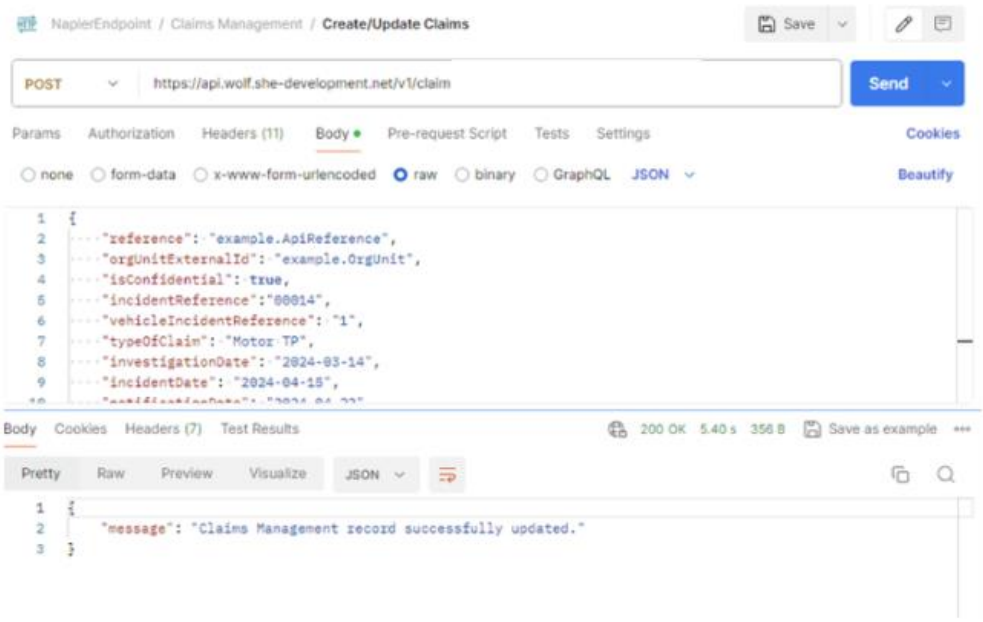
If there is no claims management records for the reference then the /v1/claims API method will create the claims management record using the details in the Claims Management JSON object (see the 'Create a claims management record' section above for details on the behaviour when creating a claims management record).

Before attempting to use the /v1/claim API method make sure you understand the claims management JSON object (see the section above) and that you have the required details to access the API (i.e. the URL, API key, etc), follow the Getting Started guide if you don't have these details already. The following table sections show how to update a claims management record using the /v1/claim API method in a variety of software tools / platforms:

Partial updates are not supported i.e. you cannot update only selected claims management fields. The claims management JSON object must contain all the fields that are required to define the records data (same as if the record was being created). Assure will update the claims management record to match the details supplied, this includes applying the default values for any fields that are not provided in the claims management JSON object. See the OpenAPI schema for details of how the claims management data will be defaulted for each field.

Any HTTP response code is the range 200-299 should be treated as success (do not use the contents of the response body for identifying success or failure as these may change). Implement effective error handling as per the guide for error handling.

EVOTIX

Postman API platform	The screenshot below shows a successful 'update claims management' request for a customer on the wolf stack (NB: the Postman setup is identical to that used for creating a Claims Management record). The API key is supplied via the 'Authorization' tab (see the Getting Started guide for how to setup the API key). The response section at the bottom shows the result Status: 200 OK which indicates that the Claims Management update was successful (also displayed is the response message from the Customer API confirming the successful update). NB: With tools like this where the Claims Management JSON object is manually entered you must ensure the contents of any string values are properly escaped. 

<table>
<tr>
<td>Windows Powershell</td>
<td>The code block below shows the few lines of PowerShell script required to setup the Claims Management Object JSON and to make the ‘update claims management’ request to the Customer API for a customer (NB: this is identical to the script for creating a claims management record). The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

<pre>
$ClaimsManagementObjectJSON = @{
    "reference"   = "exampleApiReference"
    "orgUnitExternalId":   = "example.OrgUnit"
    "incidentDate":   = "2024-04-15"
} | ConvertTo-Json
Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri https://api.wolf.sheassure.net/v1/claim `
    -Method Post `
    -ContentType 'application/json' `

    -Body $ClaimsManagementObjectJSON`
</pre>
The following shows the output from the above Powershell script code being run where a successful response is generated. The StatusCode: 200 indicates that the update a claims management record request was successful. Some response lines have been removed for brevity.

<pre>
StatusCode   : 200
StatusDescription : OK
Content   : {"message":"Claims Management record updated."}
</pre>
</td>
</tr>
<tr>
<td>Python</td>
<td>The code block below shows the few lines of Python code required to setup the Claims Management Object JSON and to make the ‘update claims management’ request to the Customer API for a customer (NB: this is identical to the script for creating a claims management record). The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

<pre>
# pip install requests
import requests
claims_management_object = {
    "reference": "exampleApiReference",
    "orgUnitExternalId": "example.OrgUnit",
    "incidentDate": "2024-04-15"
}
headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}
r = requests.post("https://api.wolf.sheassure.net/v1/claim", headers=headers, json =
claims_management_object)
r.raise_for_status()
print(f"StatusCode={r.status_code}")

print(f"Body={r.content}")
</pre>
The following shows the output from the above Python code being run where a successful response is generated. The <code>StatusCode=200</code> indicates that the claims management update was successful.

<pre>
StatusCode=200
Body=b'{"message": "Claims Management record updated."}'
</pre>
</td>
</tr>
</table>

EVOTIX

EVOTIX

Deleting a Claims Management Record

The Customer API allows the deletion of claims management records in Assure using the /v1/claim /{reference} API method with the DELETE verb. The record will be identified by its reference provided from the {reference} path parameter. The Claims Management Object is NOT required for this method.

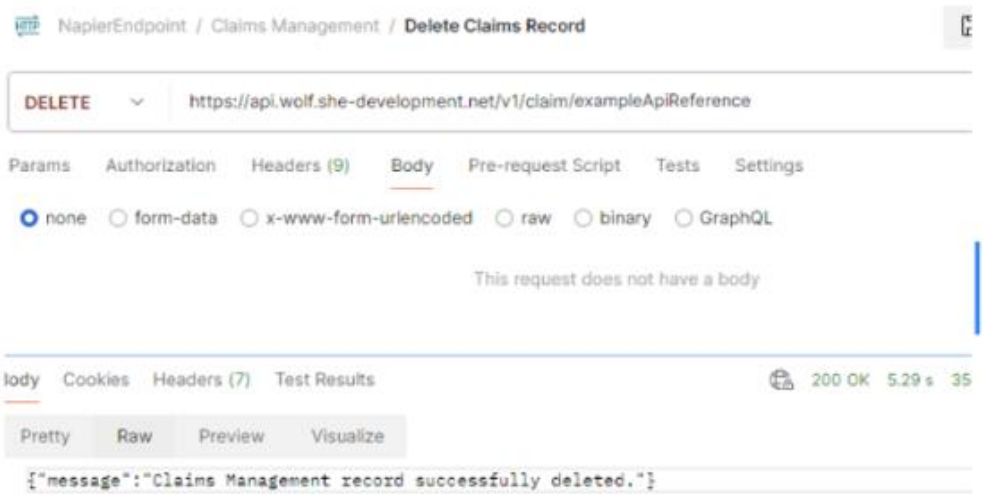
If there is no claims management record for the reference provided then the /v1/claim/{reference}API method will still return a successful response (i.e. HTTP status code in the range 200-299).

Before attempting to use the /v1/claim/{reference} API method make sure you have the required details to access the API (i.e. the URL, API key, etc), follow the Getting Started guide if you don't have these details already. The following table sections show how to delete a claims management record using the /v1/claim/{reference} API method in a variety of software tools / platforms:

Any HTTP response code is the range 200-299 should be treated as success (do not use the contents of the response body for identifying success or failure as these may change). Implement effective error handling as per the guide for error handling.

IMPORTANT - The login user name value needs to have URL escaping applied before inclusion in the URL.

EVOTIX

Postman API platform	The screenshot below shows a successful 'delete claims management record' request for a customer. Note the use of the 'Authorization' tab to configure the API key header. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX value is where the API key would be placed. The response section at the bottom shows the result Status: 200 OK which indicates that the record was successfully disabled (also displayed is the response message from the Customer API confirming the successful disable). NB: With tools like this where the Claims Management JSON object is manually entered you must ensure the contents of any string values are properly escaped. 

<table>
<tr>
<td>Windows Powershell</td>
<td>The code block below shows the few lines of Powershell script required to make the ‘delete claims management record’ request to the Customer API for a customer. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

IMPORTANT - Note the use of [System.Net.WebUtility]::UrlEncode to ensure that the reference value is properly escaped for inclusion in the URL.

<pre>$URL = "https://api.wolf.sheassure.net/v1/claim" +
[System.Net.WebUtility]::UrlEncode("example?reference")
Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri $URLEscaped `
    -Method Delete `</pre>

The following shows the output from the above Powershell script code being run where a successful response is generated. The StatusCode: 200 indicates that the disable claims management request was successful. Some response lines have been removed for brevity.

<pre>StatusCode   : 200
StatusDescription : OK
Content   : {"message": "Claims Management record successfully deleted."}</pre>
</td>
</tr>
<tr>
<td>Python</td>
<td>The code block below shows the few lines of Python code required to make the ‘delete claims management record’ request to the Customer API for a customer. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

IMPORTANT - Note the use of urllib.parse.quote to ensure that the reference value is properly escaped for inclusion in the URL.

<pre># pip install requests
# pip install urllib
import requests
import urllib
url = "https://api.wolf.sheassure.net/v1/claim/" +
urllib.parse.quote('example.reference', safe='')
headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}
r = requests.delete(url, headers=headers)
r.raise_for_status()
print(f"StatusCode={r.status_code}")

print(f"Body={r.content}")</pre>

The following shows the output from the above Python code being run where a successful response is generated. The StatusCode=200 indicates that the claims management delete was successful.

<pre>StatusCode=200
Body=b'{"message": "Claims management record successfully deleted."}'</pre>
</td>
</tr>
</table>