API Guide

Assure Customer API

User Guide: Update the Person Register via API

Evotix Ltd. Revision 4.0 (March 2025)

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 the person register in Assure. A separate guide is available for setting up the Customer API.

EVOTIX

Contents

EVOTIX

Managing Person Register Records via the Customer API

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

This allows you to create, update and delete records within the Person Register module.

Limitations

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

- The Person Register 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.
- The Person Register 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.
- It is not possible to add Project records to Person Register records in the Project Links section, via the API. Any Project records will need to be linked manually.
- Any default values set within the caption maintenance area are not taken into consideration by the API.

Pre-configuration for Person Register 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 Person Register records.

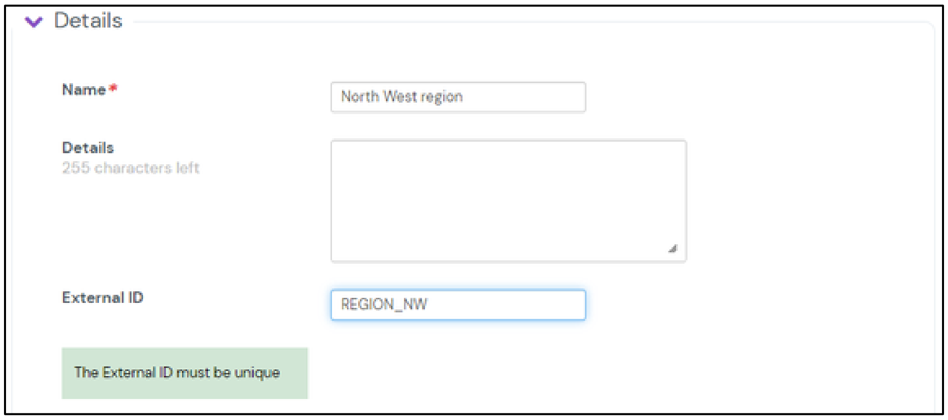
This pre-configuration is only needed for Organisational Units which have been manually added and which do not have an External ID set.

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.

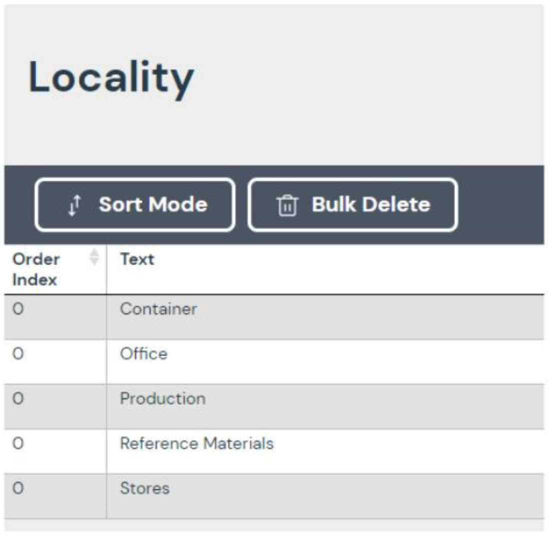


Name*	North West region
Details 255 characters left	
External ID	REGION_NW

The External ID must be unique

Picklist Values

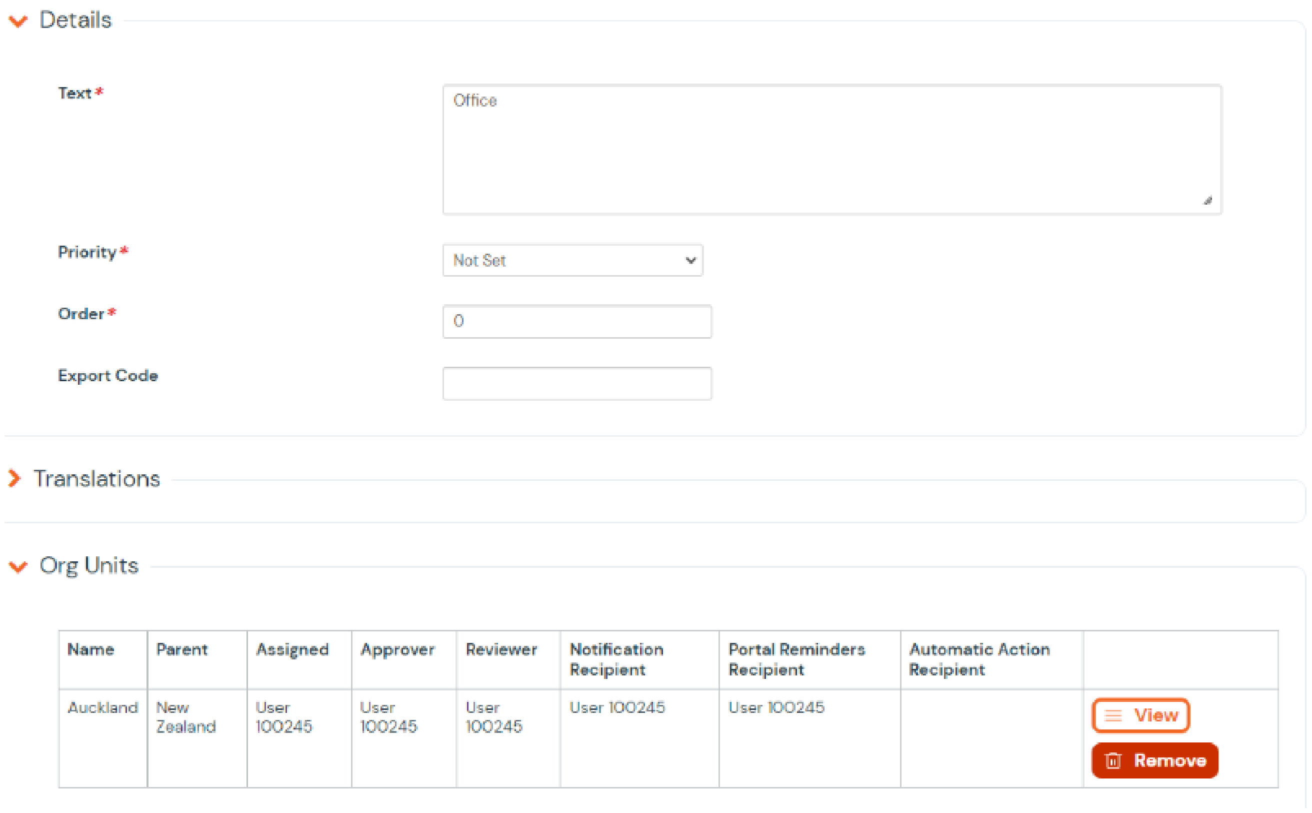
For any of the fields in a Module record that has values that come from the ‘Picklist Data Dictionary’ in Assure then the value provided in the API request must match a value in the ‘Picklist Data Dictionary' for that field and can be in the customer's default language or base system language (English-UK). 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



Order Index	Text
0	Container
0	Office
0	Production
0	Reference Materials
0	Stores

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.

EVOTIX



Details

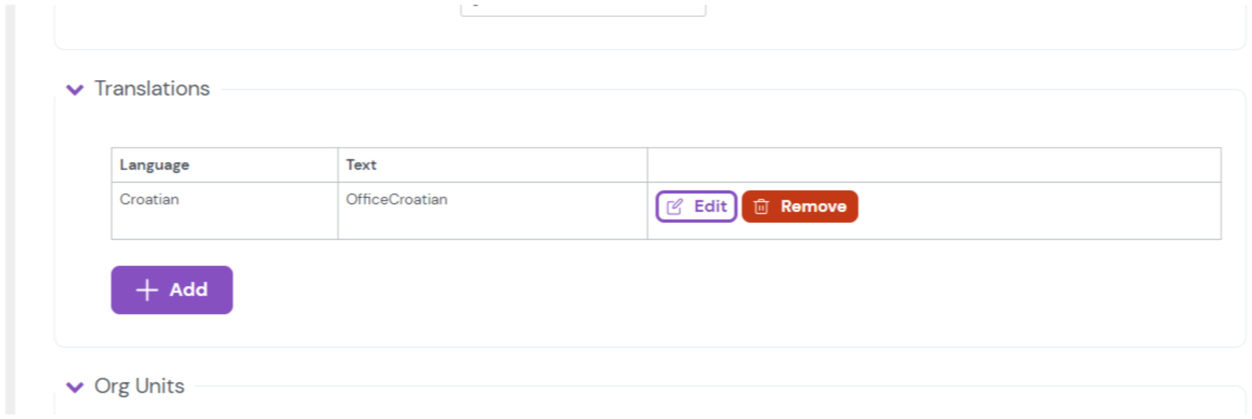
Text *	Office
Priority *	Not Set
Order *	0
Export Code	

Translations

Org Units

Name	Parent	Assigned	Approver	Reviewer	Notification Recipient	Portal Reminders Recipient	Automatic Action Recipient	
Auckland	New Zealand	User 100245	User 100245	User 100245	User 100245	User 100245		View Remove

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 APPI. 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.



Translations

Language	Text	
Croatian	OfficeCroatian	Edit Remove

Add

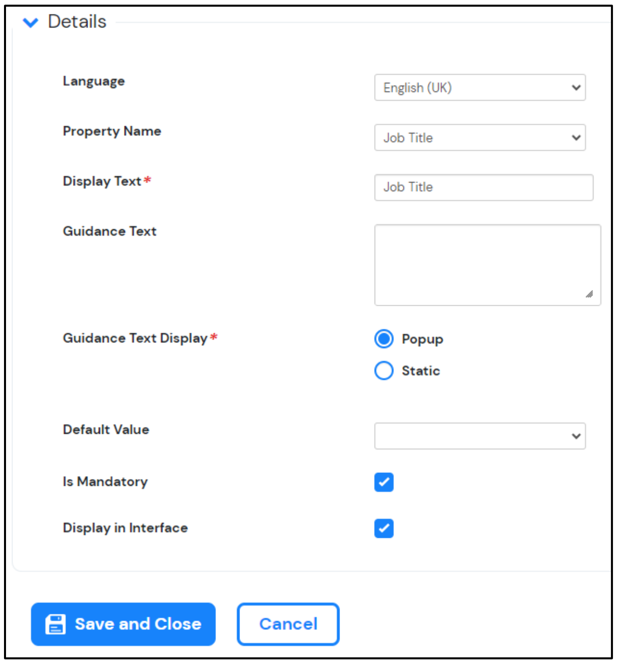
Org Units

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 Person Register 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 ‘Job Title a mandatory field. The field must also have the ‘Display in Interface’ selected for the mandatory validation to occur in the API.



Language	English (UK)
Property Name	Job Title
Display Text*	Job Title
Guidance Text	
Guidance Text Display*	[✓ Popup] Static
Default Value	
Is Mandatory	✓
Display in Interface	✓

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

Person Register JSON object

When creating and updating a Person Register record the records 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.

The OpenAPI schema for the Customer API contains the formal definition of the JSON structure i.e. the personRegisterPOSTRequestobject.

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 Person Register JSON object relate to the resulting Person Register record setup in Assure the following sections contain some worked examples.

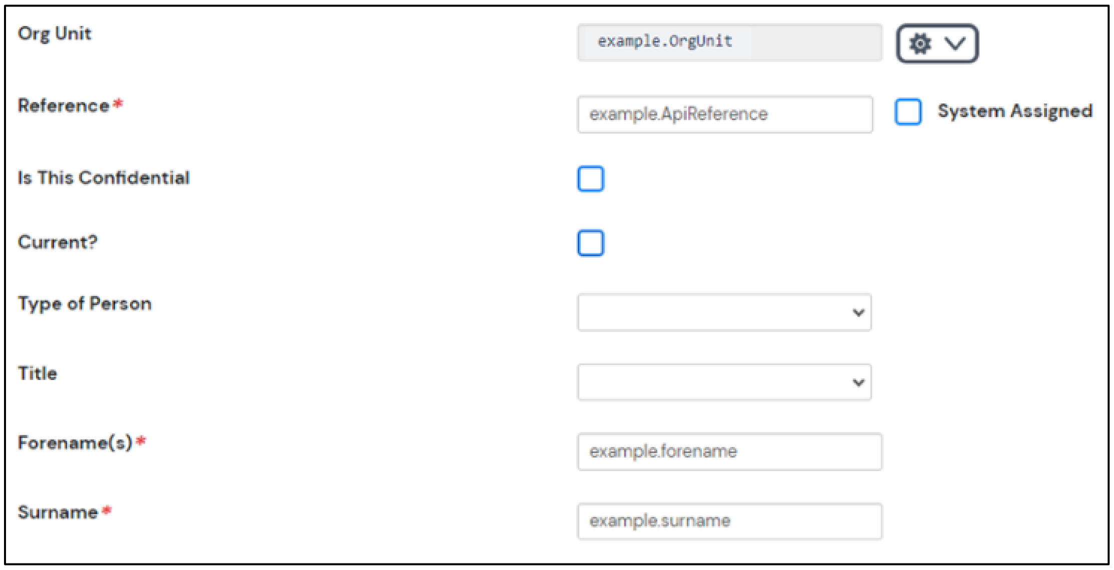
The text encoding to be used for all interactions with the Customer API is UTF-8. This is pretty much the standard today for software tools and platforms however it is important to check that you are using UTF-8 as if not then when you get foreign characters in the data or other symbols like emoticons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

Minimum Person Register data

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

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit"
}
```



Org Unit	example.OrgUnit
Reference*	example.ApiReference ☐ System Assigned
Is This Confidential	☐
Current?	☐
Type of Person	
Title	
Forename(s)*	example.forename
Surname*	example.surname

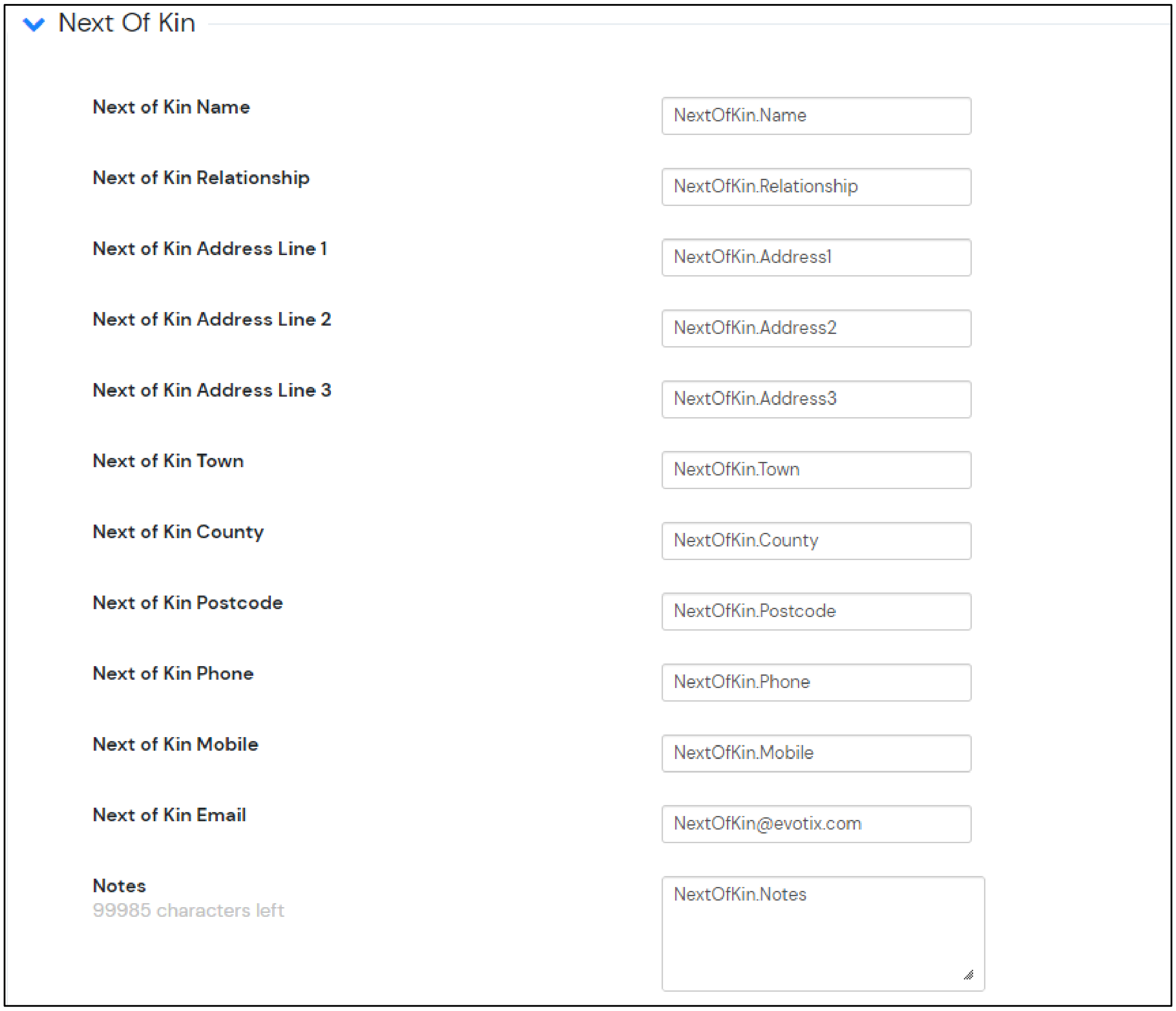
Further Person Register details omitted for brevity as they will all be the defaults.

Person Register record with next of kin details

This example shows the same Person Register record with next of kin details.

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit",
    "nextOfKin":
        {
            "name": "NextOfKin.Name",
            "address1" : "NextOfKin.Address1",
            "address2" : "NextOfKin.Address2",
            "address3" : "NextOfKin.Address3",
            "town" : "NextOfKin.Town",
            "county": "NextOfKin.County",
            "postcode": "NextOfKin.Postcode",
            "phone" : "NextOfKin.Phone",
            "relationship": "NextOfKin.Relationship",
            "mobile" : "NextOfKin.Mobile",
            "email": "NextOfKin@evotix.com",
            "notes" : "NextOfKin.Notes",
            "additionalfield1" : "additionalField1",
            "additionalfield2" : "additionalField2"
        }
}
```

The Person Register Record details will be the same as the 'minimum Person Register data' screen shot above. The difference in this case is that the next of kin details are included in the Next of Kin section.


Next Of Kin

Next of Kin Name	NextOfKin.Name
Next of Kin Relationship	NextOfKin.Relationship
Next of Kin Address Line 1	NextOfKin.Address1
Next of Kin Address Line 2	NextOfKin.Address2
Next of Kin Address Line 3	NextOfKin.Address3
Next of Kin Town	NextOfKin.Town
Next of Kin County	NextOfKin.County
Next of Kin Postcode	NextOfKin.Postcode
Next of Kin Phone	NextOfKin.Phone
Next of Kin Mobile	NextOfKin.Mobile
Next of Kin Email	NextOfKin@evotix.com
Notes 99985 characters left	NextOfKin.Notes

EVOTIX

Person Register record with linked user

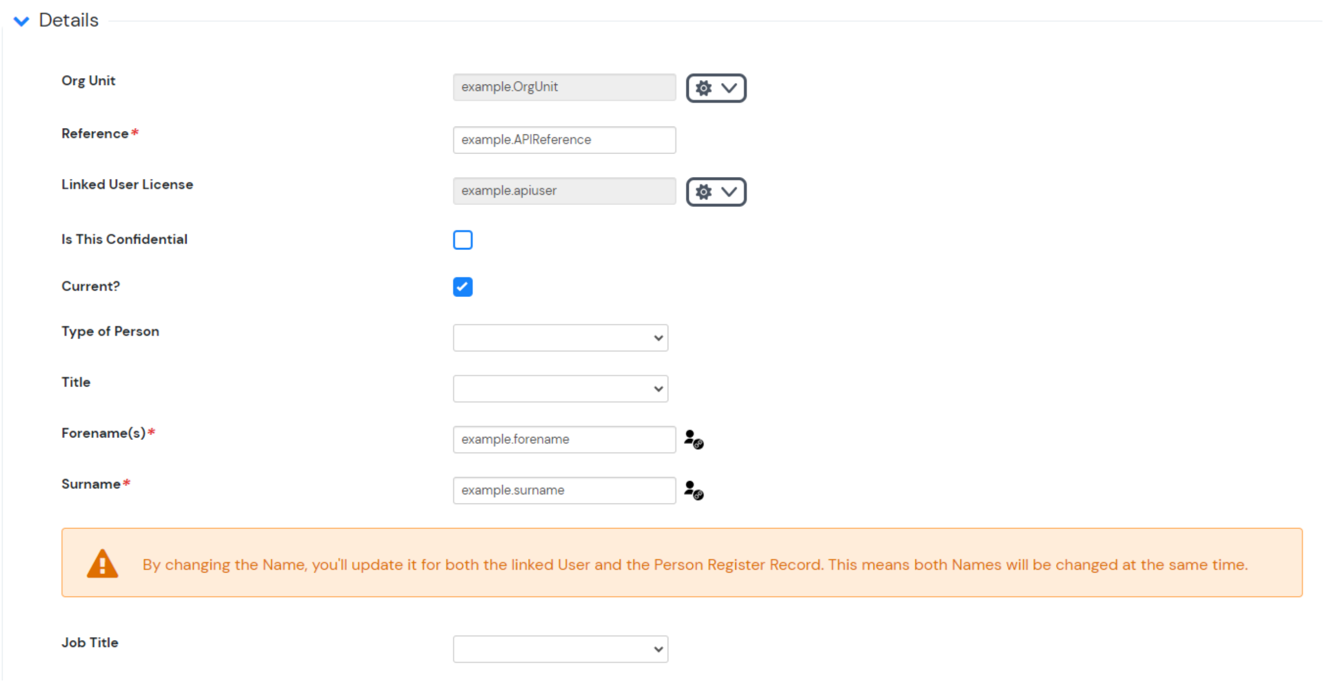
This example shows the same Person Register record with a linked user.

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit",
    "linkedUsername": "example.apiuser",
    "email" : "example.apiuser@evotix.com",
}
```

In order to link a Person Register record with a User, you must have the feature enabled in System Settings.

In order to link a User to a Person Register record, the request will include the same fields as in the ‘minimum person register data’ section, but it will also require the ‘linkedUsername' field and in the 'email’ field. The email must be a unique email.

Note: the values in the email and name fields (forename and surname) values will also update the linked Users email and full name to match.

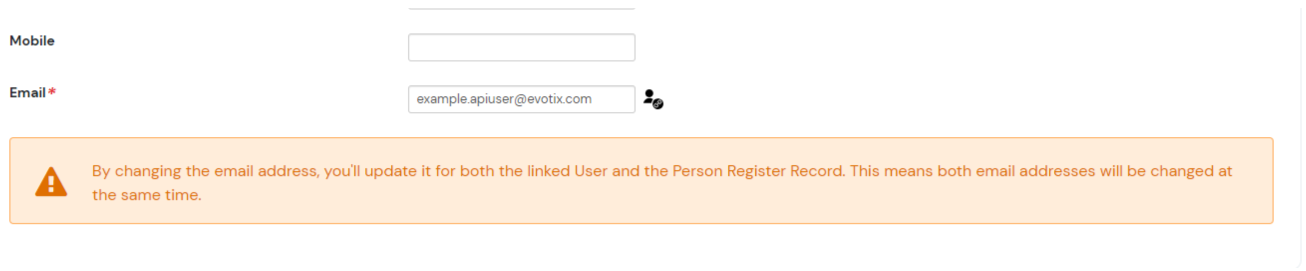

Details

Org Unit	example.OrgUnit
Reference*	example.APIReference
Linked User License	example.apiuser
Is This Confidential	☐
Current?	✓
Type of Person	
Title	
Forename(s)*	example.forename
Surname*	example.surname

By changing the Name, you'll update it for both the linked User and the Person Register Record. This means both Names will be changed at the same time.

Job Title	

EVOTIX


Mobile

Email *

example.apiuser@evotix.com

By changing the email address, you'll update it for both the linked User and the Person Register Record. This means both email addresses will be changed at the same time.

Person Register record with training

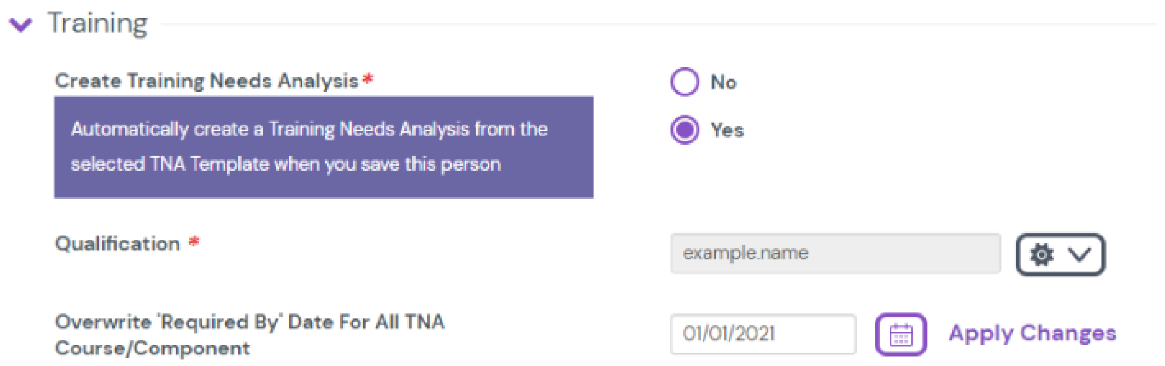
This example shows the same Person Register record with the training section defined.

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit",
    "training" : {
        "createTrainingNeedsAnalysis" : true,
        "trainingNeedsAnalysisReference" : "example.tnaReference",
        "overwriteRenewedByDate": "2021-01-01"
    }
}
```

The training section allows for a Training Needs Analysis record (TNA) to be created on the creation of a Person Register Record via the post endpoint. This section is only relevant on creation of a person record. It is not possible to create a TNA record when editing a person record, nor is it possible to edit the record from this endpoint. If you provide this section in an update request and set ‘createTrainingNeedsAnalysis' to be 'true’ an error will be returned.

The training and TNA Template modules must be enabled in order to create training records via this endpoint.

To create a training record, the TNA template reference provided in the 'trainingNeedsAnalysisReference' field must be unique. It is not possible to create a training record when the reference provided belongs to multiple TNA templates. note. In the UI the ‘Qualification’ field has the course name and not the reference displayed. For the API you must provide the reference.



Training

Create Training Needs Analysis *

Automatically create a Training Needs Analysis from the selected TNA Template when you save this person

No [✓ Yes]

Qualification * example.name

Overwrite 'Required By' Date For All TNA Course/Component 01/01/2021

EVOTIX

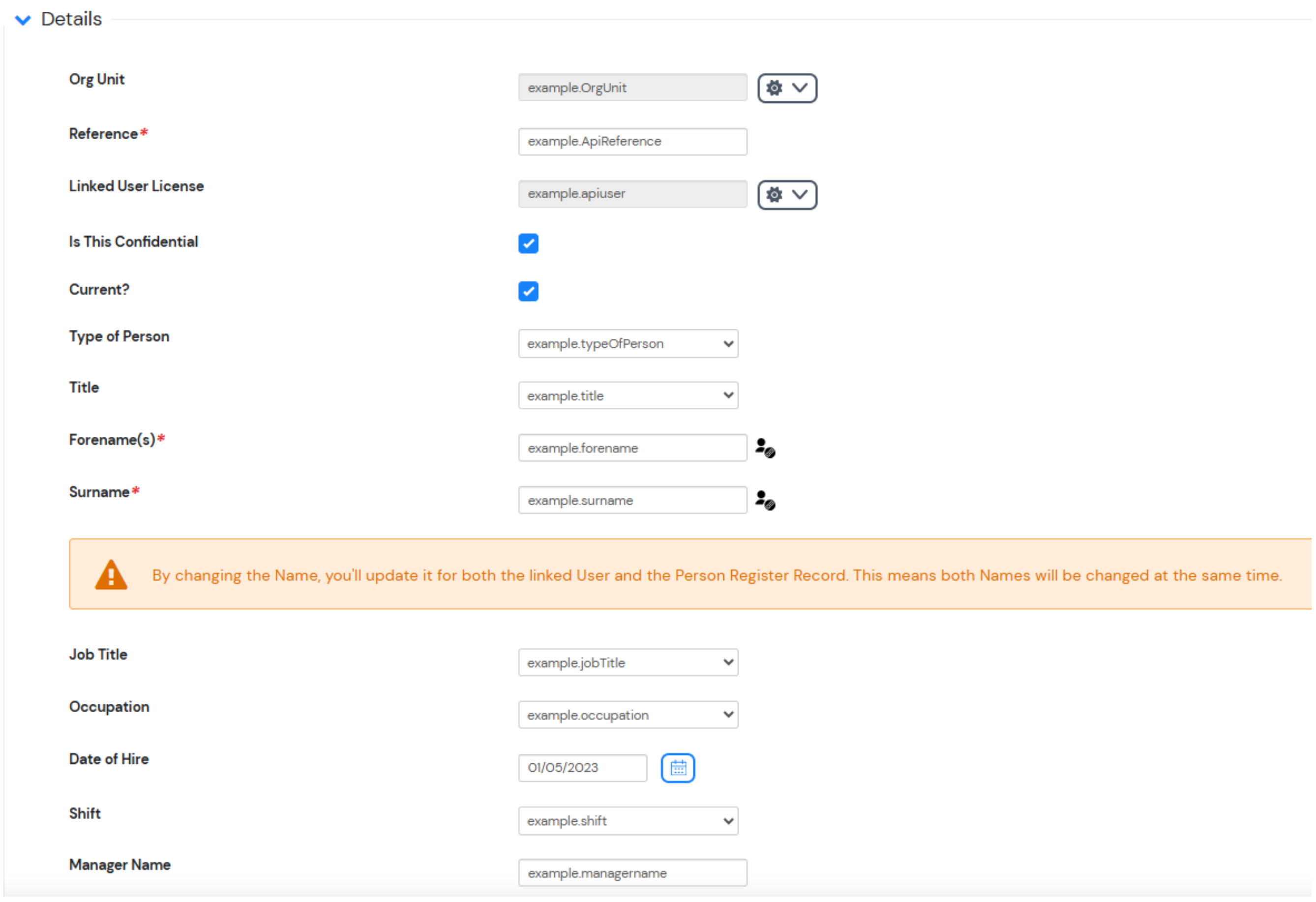
Person Register record with everything defined

This example shows the same Person Register record with all the possible fields defined.

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit",
    "linkedUsername": "example.apiuser",
    "email" : "evotix@evotix.com",
    "iscurrent" : true,
    "DateOfBirth" : "1998-01-01",
    "Title" : "example.Title",
    "JobTitle" : "example.JobTitle",
    "isConfidential": true,
    "maritalStatus": "example.maritalStatus",
    "niNumber": "example.niNumber",
    "nationality": "example.nationality",
    "dateOfHire": "2023-05-01",
    "shift": "example.shift",
    "occupation": "example.occupation",
    "phone": "example.ManagerPhone",
    "mobile": "example.ManagerMobile",
    "typeOfPerson": "example.typeOfPerson",
    "managerName": "example.managerName",
    "managerPhone": "example.managerPhone",
    "ageRange": "example.ageRange",
    "address1": "example.address1",
    "address2": "example.address2",
    "address3": "example.address3",
    "town" : "example.town",
    "county": "example.county",
    "postcode": "example.postcode",
    "location": "example.location",
    "locality" : "example.locality",
    "nextOfKin":
        {
            "name": "NextOfKin.Name",
            "address1" : "NextOfKin.Address1",
            "address2" : "NextOfKin.Address2",
            "address3" : "NextOfKin.Address3",
            "town" : "NextOfKin.Town",
            "county": "NextOfKin.County",
            "postcode": "NextOfKin.Postcode",
            "phone" : "NextOfKin.Phone",
            "relationship": "NextOfKin.Relationship",
            "mobile" : "NextOfKin.Mobile",
            "email": "NextOfKin@evotix.com",
            "notes" : "NextOfKin.Notes",
            "additionalfield1" : "additionalField1",
            "additionalfield2" : "additionalField2"
        }
}
```

It is not necessary to set all the fields when creating or updating a Person Register 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.

The Person Register record next of kin details will be the same as the screenshot above.

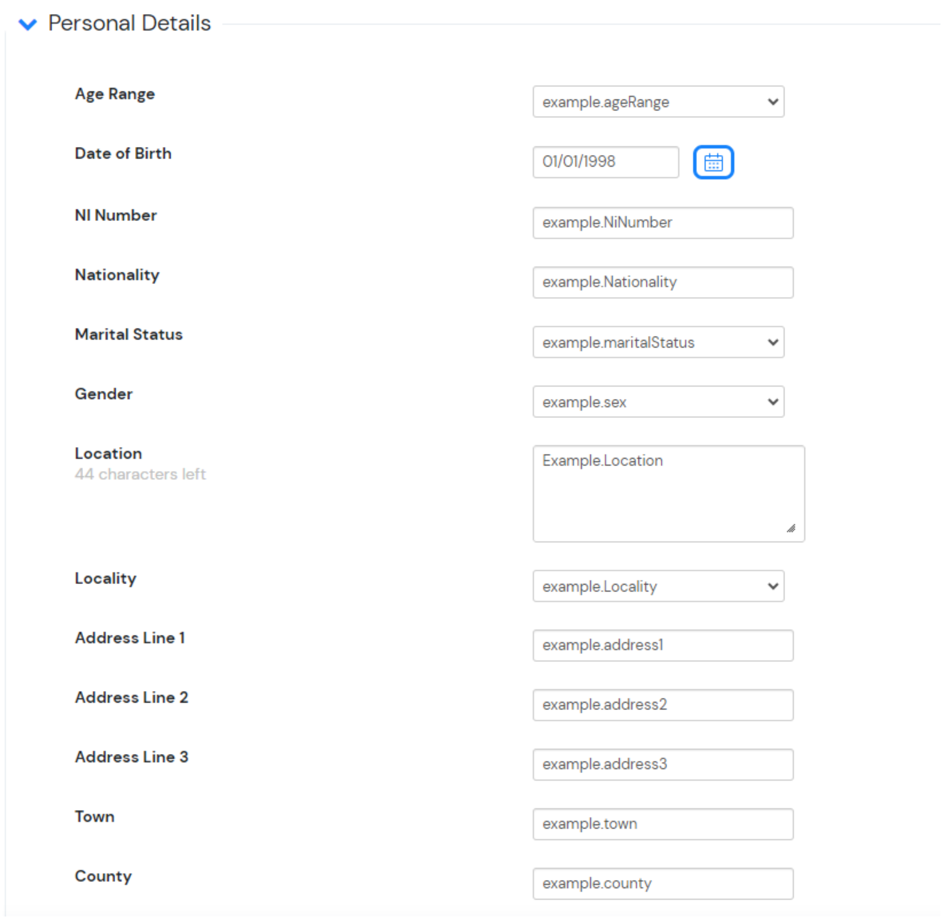


Org Unit	example.OrgUnit
Reference*	example.ApiReference
Linked User License	example.apiuser
Is This Confidential	✓
Current?	✓
Type of Person	example.typeOfPerson
Title	example.title
Forename(s)*	example.forename
Surname*	example.surname

By changing the Name, you'll update it for both the linked User and the Person Register Record. This means both Names will be changed at the same time.

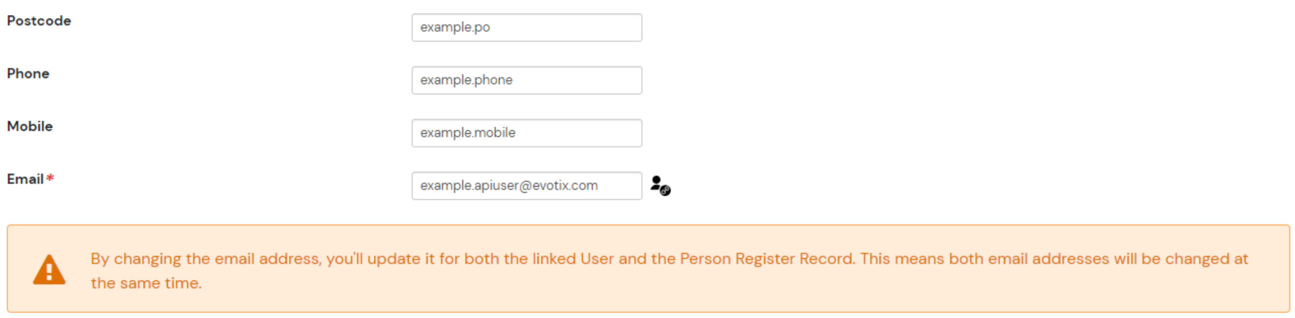
Job Title	example.jobTitle
Occupation	example.occupation
Date of Hire	01/05/2023
Shift	example.shift
Manager Name	example.managername

EVOTIX



Personal Details

Age Range	example.ageRange
Date of Birth	01/01/1998
NI Number	example.NiNumber
Nationality	example.Nationality
Marital Status	example.maritalStatus
Gender	example.sex
Location 44 characters left	Example.Location
Locality	example.Locality
Address Line 1	example.address1
Address Line 2	example.address2
Address Line 3	example.address3
Town	example.town
County	example.county



Postcode	example.po
Phone	example.phone
Mobile	example.mobile
Email*	example.apiuser@evotix.com

By changing the email address, you'll update it for both the linked User and the Person Register Record. This means both email addresses will be changed at the same time.

EVOTIX

Person Register JSON PATCH object

When sending a patch request to update a person register record details they 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 personRegisterPATCHRequestobject. 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.

The below section of JSON shows all the fields in the Person Register JSON PATCH Object.

Note. The object contains almost all the same fields as the JSON POST object but it does not contain the reference field in the body or the training section. It does not include the training section as a training record cannot be edited or created on update of a Person Register record.

EVOTIX

```
{
    "forename": "example.forename",
    "surname": "example.surname",
    "reference" : "example.ApiReference",
    "orgUnitExternalId" : "example.OrgUnit",
    "linkedUsername": "example.apiuser",
    "email" : "evotix@evotix.com",
    "iscurrent" : true,
    "DateOfBirth" : "1998-01-01",
    "Title" : "example.Title",
    "JobTitle" : "example.JobTitle",
    "isConfidential": true,
    "maritalStatus": "example.maritalStatus",
    "niNumber": "example.niNumber",
    "nationality": "example.nationality",
    "shift": "example.shift",
    "dateOfHire": "2023-05-01",
    "occupation": "example.occupation",
    "phone": "example.ManagerPhone",
    "mobile": "example.ManagerMobile",
    "typeOfPerson": "example.typeOfPerson",
    "managerName": "example.managerName",
    "managerPhone": "example.managerPhone",
    "ageRange": "example.ageRange",
    "address1": "example.address1",
    "address2": "example.address2",
    "address3": "example.address3",
    "town" : "example.town",
    "county": "example.county",
    "postcode": "example.postcode",
    "location": "example.location",
    "locality" : "example.locality",
    "nextOfKin":
        {
            "name": "NextOfKin.Name",
            "address1" : "NextOfKin.Address1",
            "address2" : "NextOfKin.Address2",
            "address3" : "NextOfKin.Address3",
            "town" : "NextOfKin.Town",
            "county": "NextOfKin.County",
            "postcode": "NextOfKin.Postcode",
            "phone" : "NextOfKin.Phone",
            "relationship": "NextOfKin.Relationship",
            "mobile" : "NextOfKin.Mobile",
            "email": "NextOfKin@evotix.com",
            "notes" : "NextOfKin.Notes",
            "additionalfield1" : "additionalField1",
            "additionalfield2" : "additionalField2"
        }
}
```

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

EVOTIX

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.

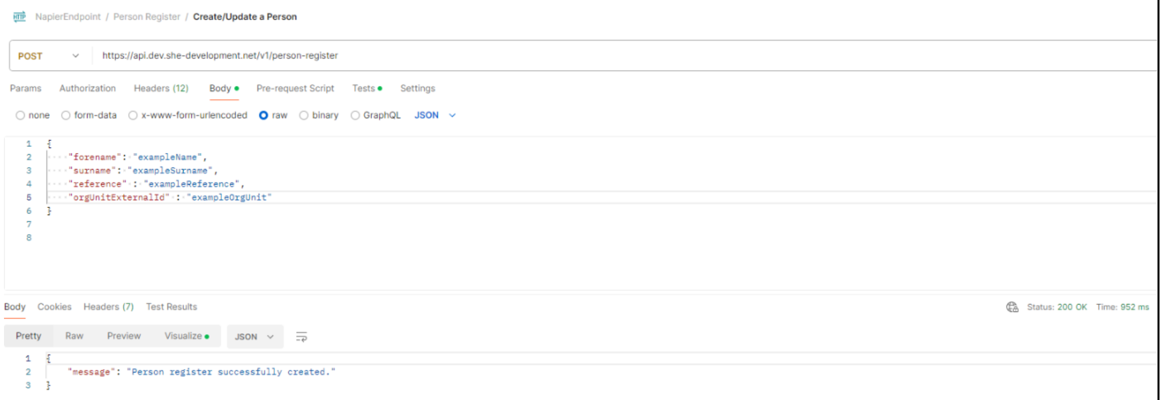
Creating a Person Register Record

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

If there is an existing person register record for the external ID then the /v1/person-register API method will update the existing Person Register record to match the supplied details (see the 'Update a Person Register Record' section below for details on the behaviour when updating an existing Person Register record). Note: If there are multiple person register 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/person-register API method make sure you understand the Person Register 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 person register record using the /v1/person-register 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.

Postman API platform	The screenshot below shows a successful 'create person register record' request for a customer on the uk2 stack. 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 Person Register JSON POST object is manually entered you must ensure the contents of any string values are properly escaped. 

NapierEndpoint / Person Register / **Create/Update a Person**

POST https://api.dev.she-development.net/v1/person-register

```
{
    "forename": "exampleName",
    "surname": "exampleSurname",
    "reference" : "exampleReference",
    "orgUnitExternalId" : "exampleOrgUnit"
}
```

Status: 200 OK Time: 952 ms

```
{
    "message": "Person register successfully created."
}
```

Windows Powershell

The code block below shows the few lines of Powershell script required to setup the Person Register Object JSON POST and to make the ‘create person register record’ request to the Customer API for a customer on the uk2 stack. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

```
$PersonRegisterObjectJSON = @{
    "forename"                  = "example.forename",
    "surname"                   = "example.surname",
    "reference"                 = "example.ApiReference",
    "orgUnitExternalId"         = "example.OrgUnit"
} | ConvertTo-Json

Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri https://api.uk2.sheassure.net/v1/person-register `
    -Method Post `
    -ContentType 'application/json' `
    -Body $PersonRegisterObjectJSON`
```

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 person register request was successful. Some response lines have been removed for brevity.

```
StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Person register successfully created."}
```

EVOTIX

Python	The code block below shows the few lines of Python code required to setup the Person Register Object JSON POST and to make the 'create person register' request to the Customer API for a customer on the uk2 stack. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.
	
	`# pip install requests`
	`import requests`
	
	`person_register_object = {`
	`    "forename": "example.forename",`
	`    "surname": "example.surname",`
	`    "reference" : "example.ApiReference",`
	`    "orgUnitExternalId" : "example.OrgUnit"`
	`}`
	
	`headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}`
	`r = requests.post("https://api.uk2.sheassure.net/v1/person-register", headers=headers, json = person_register_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 person register record creation was successful.
	
	`StatusCode=200`
	`Body=b'{"message": "Person register successfully created."}'`

Updating a Person Register Record with POST request

The Customer API allows the updating of person register records in Assure using the /v1/person-register API method with the POST verb. The person register record details are supplied in the body of the API request using the Person Register JSON POST object. Person register records are identified by their reference, which is the value in the reference field of the Person Register JSON POST object. If there is an existing person register record for the reference provided then the /v1/person-register API method will update the existing person register record to match the supplied details (see the 'Update a person register record' section below for details on the behaviour when updating an existing person register record). Note: If there are multiple person register 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 person register records for the reference then the /v1/person-register API method will create the person register record using the details in the Person Register JSON POST object (see the 'Create a person register record' section above for details on the behaviour when creating a person register record).

Before attempting to use the /v1/person-register API method make sure you understand the Person Register JSON POST 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 person register record using the /v1/person-register API method in a variety of software tools / platforms:

Partial updates are not supported with a POST request. If you wish to perform a Partial Update please see Managing Person Register Records via the API | Update a person register record with PATCH request section.

If you try to update a record with using the POST request Assure will update the person register record to match the details supplied, this includes applying the default values for any fields that are not provided in the Person Register JSON POST object. See the OpenAPI schema for details of how the person register data will be defaulted for each field.

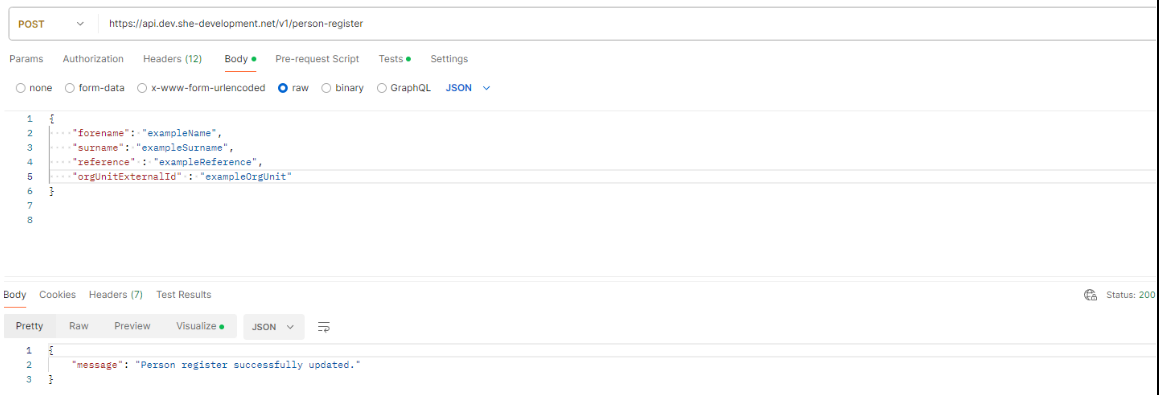
Any HTTP response code is the range 200-299 should be treated as success (do not use the contents of the response body for identifying success or failure as these may change). Implement effective error handling as per the guide for error handling.

EVOTIX

Postman API platform

The screenshot below shows a successful 'update person register' request for a customer on the uk2 stack (NB: the Postman setup is identical to that used for creating a person register 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 person register update was successful (also displayed is the response message from the Customer API confirming the successful update).

NB: With tools like this where the Person Register JSON object is manually entered you must ensure the contents of any string values are properly escaped.



https://api.dev.she-development.net/v1/person-register

```
{
    "forename": "exampleName",
    "surname": "exampleSurname",
    "reference" : "exampleReference",
    "orgUnitExternalId" : "exampleOrgUnit"
}
```

Status: 200

```
{
    "message": "Person register successfully updated."
}
```

Windows Powershell

The code block below shows the few lines of Powershell script required to setup the Person Register JSON POST object and to make the 'update person register' request to the Customer API for a customer on the uk2 stack (NB: this is identical to the script for creating a person register record). The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.

```
$PersonRegisterObjectJSON = @{
    "forename"                = "example.forename",
    "surname"                 = "example.surname",
    "reference"               = "example.ApiReference",
    "orgUnitExternalId"       = "example.OrgUnit"
} | ConvertTo-Json

Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri https://api.uk2.sheassure.net/v1/person-register `
    -Method Post `
    -ContentType 'application/json' `
    -Body $PersonRegisterObjectJSON`
```

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 person register record request was successful. Some response lines have been removed for brevity.

```
StatusCode        : 200
StatusDescription : OK
Content           : {"message":"Person register updated."}
```

Python	The code block below shows the few lines of Python code required to setup the Person Register JSON POST object and to make the ‘update person register’ request to the Customer API for a customer on the uk2 stack (NB: this is identical to the script for creating a person register record). The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.
	
	`# pip install requests`
	`import requests`
	
	`person_register_object = {`
	`    "forename": "example.forename",`
	`    "surname": "example.surname",`
	`    "reference" : "example.ApiReference",`
	`    "orgUnitExternalId" : "example.OrgUnit"`
	`}`
	
	`headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}`
	`r = requests.post("https://api.uk2.sheassure.net/v1/person-register", headers=headers, json = person_register_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 person register update was successful.
	
	`StatusCode=200`
	`Body=b'{"message": "Person register updated."}'`

Partial update of a Person Register Record with PATCH request

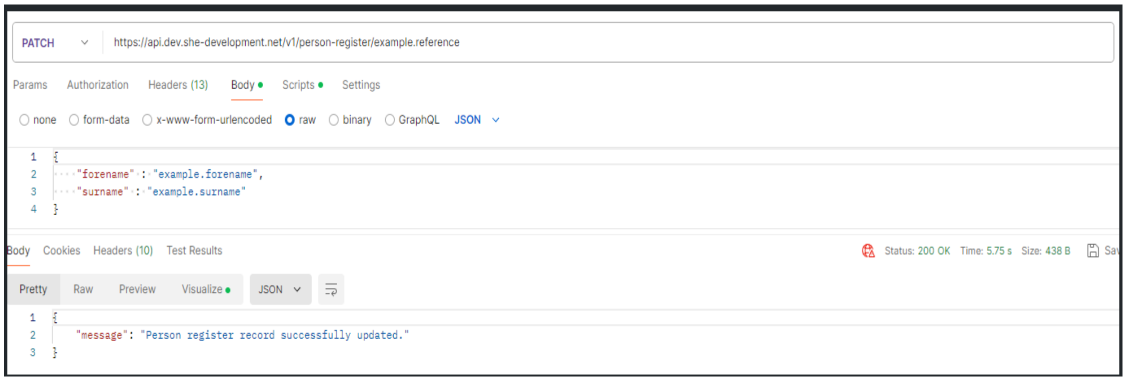
The Customer API allows the partial update of person register records in Assure using the /v1/person-register/{reference} API method with the PATCH verb. A PATCH request will only update the fields provided in the request and any other fields will remain unchanged.

The record that will be updated will be identified by its reference provided from the {reference} path parameter. The person register record details to be updated are supplied in the body of the API request using the person register JSON PATCH object. If there is an existing person register record for the reference provided then the /v1/person-register/{reference}API method will update the existing person register record with the supplied details. Note: If there are multiple person register records with the same reference that is provided in the request this will result in an error response (HTTP error 409). Additionally if there is no person register record with the reference that is provided in the request this will result in an error response (HTTP error 404).

Before attempting to use the /v1/person-register/{reference} API method make sure you understand the person register JSON PATCH 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 perform a partial update a person register record using the /v1/person-register/{reference} API method in a variety of software tools / platforms.

Note. All examples shown below are for a partial update of the 'forename' and 'surname', any field in the JSON PATCH object can also be used for a partial update.

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.

Postman API platform	The screenshot below shows a successful 'partial update of a person register record' request for a customer on the uk2 stack (NB: the Postman setup is identical to that used for creating a person register 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 person register update was successful (also displayed is the response message from the Customer API confirming the successful update).
	
	PATCH https://api.dev.she-development.net/v1/person-register/example.reference
	Body: raw JSON
	`{`
	`    "forename" : "example.forename",`
	`    "surname" : "example.surname"`
	`}`
	Status: 200 OK Time: 5.75 s Size: 438 B
	`{`
	`    "message": "Person register record successfully updated."`
	`}`
	 Params Authorization Headers (13) Body Scripts Settings
	none form-data x-www-form-urlencoded raw binary GraphQL JSON
	Body Cookies Headers (10) Test Results
	Pretty Raw Preview Visualize JSON
	NB: With tools like this where the Person Register JSON Patch object is manually entered you must ensure the contents of any string values are properly escaped.

Windows Powershell	The code block below shows the few lines of Powershell script required to setup a partial update using the Person Register Object Patch JSON. It also shows how to make the Patch partial update request to the Customer API for a customer on the uk2 stack. 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.
	
	`$PersonRegisterObjectJSON = @{`
	`    "forename"                 = "example.forename",`
	`    "surname"                  = "example.surname",`
	`} | ConvertTo-Json`
	``Invoke-WebRequest ` ``
	``    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} ` ``
	`    -Uri https://api.uk2.sheassure.net/v1/person-register/ +`
	`[System.Net.WebUtility]::UrlEncode("example?reference")`
	``    -Method Patch ` ``
	``    -ContentType 'application/json' ` ``
	``    -Body $PersonRegisterObjectJSON` ``
	
	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 partial update of the person register record request was successful. Some response lines have been removed for brevity.
	
	`StatusCode        : 200`
	`StatusDescription : OK`
	`Content           : {"message":"Person register updated."}`

Python	The code block below shows the few lines of Python code required to setup the Person Register JSON PATCH Object and to make the 'update person register' request to the Customer API for a customer on the uk2 stack (NB: this is identical to the script for creating a person register record). The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX is where the API key needs to be placed.
	
	`# pip install requests`
	
	`# pip install urllib`
	
	`import requests`
	
	`import urllib`
	
	`person_register_object = {`
	
	`    "forename": "example.forename",`
	
	`    "surname": "example.surname",`
	
	`}`
	
	`headers = { "x-api-key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"}`
	
	`url = "https://api.uk2.sheassure.net/v1/person-register/" + urllib.parse.quote('example.reference', safe='')`
	
	`r = requests.patch(url, headers=headers, json = person_register_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 person register partial update was successful.
	
	`StatusCode=200`
	`Body=b'{"message": "Person register updated."}'`

Deleting a Person Register Record

The Customer API allows the deletion of person register records in Assure using the /v1/person-register/{reference} API method with the DELETE verb. The record will be identified by its reference provided from the {reference} path parameter. The Person Register Object is NOT required for this method.

If there is no person register record for the reference provided then the /v1/person-register/{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/person-register/{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 person register record using the /v1/person-register/{reference} API method in a variety of software tools / platforms:

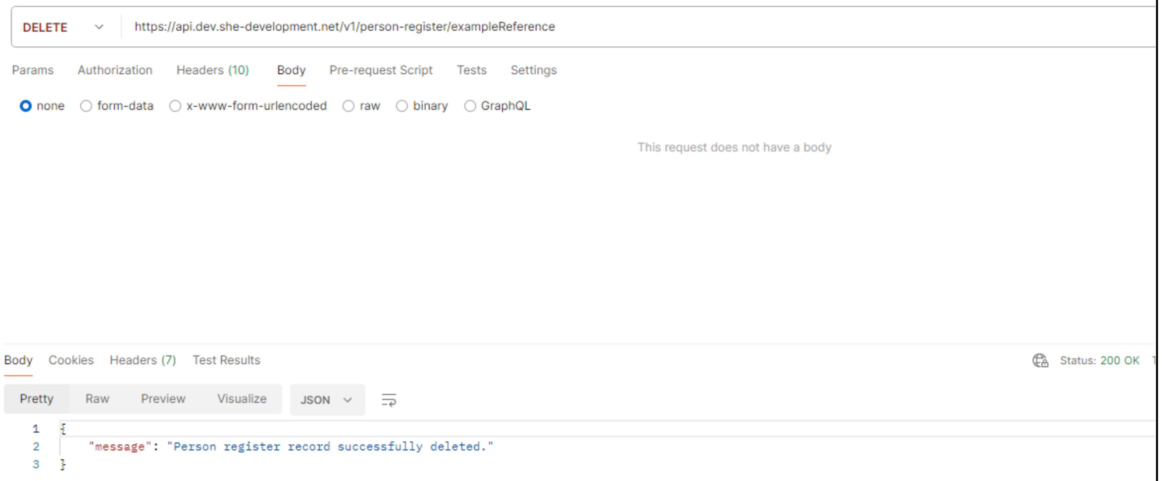
Any HTTP response code is the range 200-299 should be treated as success (do not use the contents of the response body for identifying success or failure as these may change). Implement effective error handling as per the guide for error handling.

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

Postman API platform

The screenshot below shows a successful 'delete person register record' request for a customer on the uk2 stack. Note the use of the 'Authorization' tab to configure the API key header. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 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 reference to be deleted is manually entered you must ensure that URL escaping is applied to the reference.



```
DELETE https://api.dev.she-development.net/v1/person-register/exampleReference
```

This request does not have a body

Status: 200 OK

```
{
    "message": "Person register record successfully deleted."
}
```

Windows Powershell

The code block below shows the few lines of Powershell script required to make the 'delete person register record' request to the Customer API for a customer on the uk2 stack. The XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 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.

```
$URL = "https://api.uk2.sheassure.net/v1/person-register/" +
[System.Net.WebUtility]::UrlEncode("example?reference")
Invoke-WebRequest `
    -Headers @{'x-api-key' = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'} `
    -Uri $URLEscaped `
    -Method Delete `
```

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 person register request was successful. Some response lines have been removed for brevity.

```
StatusCode        : 200
StatusDescription : OK
Content           : {"message": "Person register record successfully deleted."}
```

EVOTIX

Python	The code block below shows the few lines of Python code required to make the 'delete person register record' request to the Customer API for a customer on the uk2 stack. 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.
	
	`# pip install requests`
	`# pip install urllib`
	`import requests`
	`import urllib`
	
	`url = "https://api.uk2.sheassure.net/v1/person-register/" +`
	`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}")`
	
	The following shows the output from the above Python code being run where a successful response is generated. The StatusCode=200 indicates that the person register delete was successful.
	
	`StatusCode=200`
	`Body=b'{"message": "Person register record successfully deleted."}'`