GoodID

Scopes and claims

Version: v0.3

February 2017

Requesting information about the authenticated end-user is one of the aspects of GoodID. Say you're running a webshop. You would need the shipping and billing address as well as the email of your end-users. These are all provided by your end-users by creating one or more profiles on their GoodID apps.

1 Definition

There are two concepts that will provide you the data you need: **user claims** and **scopes**.

A **user claim** is a piece of information about your end-user. A user claim can for example be the email address, the ZIP code or place of birth.

User claims can either be voluntary or mandatory. Your business strategy will dictate which are the ones that you absolutely must have, and the ones that your users might fill out later or are totally optional. As a general rule of thumb, please require as few mandatory claims as possible because when you request a lot of information, it might turn down your end-users.

Scopes are used to require info as well. Think of scopes as a grouping of user claims. When you require a scope, you will tell that you want a specific set of user claims without having to specify which, one by one.

There are two parameters that can be used to tell which claims are requested: the `scope` and `claims` parameters of the Authorization Endpoint. While the `scope` parameter allows you to define requested user claims in one go, it doesn't let you customize the retrieval place and optionality; for this, you use the `claims` parameter where you can define where you want these claims (UserInfo) and whether a claim is mandatory or optional.
Please see the detailed description of the possible values and formatting of these parameters in chapter 2.1.1 of the GoodID Technical reference.

2 List of user claims

The following is a list of user claims that can be added to the `claims` Authorization Endpoint parameter:

2.1 Personal information

• prefix	Name prefix
• given_name	First name
• family_name	Last name
• middle_name	Middle name
• name	Full name
• gender	Gender
• birthdate	Birth date
• email	Email address
• email_verified	Email verification status
• phone_number	Phone number
• phone_number_verified	Phone verification status
• website	Website, homepage
• nickname	Nickname
• preferred_username	A username preference, optionally taken into account
• picture_data	Avatar of user in a base64 data format
• picture	Url of avatar

- locale — Locale
- zoneinfo — Timezone info

The following phone_number_parts claim gives back the detalied structure of phone_number claim:

- phone_number_parts.country_code — Phone number country code part
- phone_number_parts.number — Phone number part

2.2 Address information

- address — Address structure – see below
- address.country — Country of address
- address.country_code_iso_2 — ISO 3166-1 alpha-2 representation of the country
- address.street_address — Street address
- address.locality — City
- address.district — District
- address.postal_code — ZIP code
- address.region — State, region
- address.formatted — Formatted address, comprised of the above address fields

The following address.street_address_parts claim gives back the detalied structure of address.street_address claim:

- address.street_address_parts.street — Street
- address.street_address_parts.house_number — House number
- address.street_address_parts.building — Building
- address.street_address_parts.floor — Floor
- address.street_address_parts.door — Door
- address.street_address_parts.doorbell — Doorbell

2.3 Billing address

- billto.prefix — Name prefix
- billto.name — Full Name
- billto.given_name — First name
- billto.family_name — Last name
- billto.middle_name — Middle name
- billto.company_name — Billing company name
- billto.email — Billing contact name
- billto.phone_number — Billing contact phone
- billto.tax_id — Tax ID
- billto.email — Email
- billto.phone_number — Phone number
- billto.address — Address structure – see below
- billto.address.street_address — Street address
- billto.address.locality — City
- billto.address.district — District
- billto.address.postal_code — ZIP code
- billto.address.region — State
- billto.address.country — Country
- billto.address.country_code_iso_2 — ISO 3166-1 alpha-2 representation of the country
- billto.address.formatted — Formatted billing address, consists of the address fields above

The following billto.address.street_address_parts claim gives back the detalied structure of billto.address.street_address claim:

- billto.address.street_address_parts.street Detailed address street Street
- billto.address.street_address_parts.house_number House number
- billto.address.street_address_parts.building Building
- billto.address.street_address_parts.floor Floor
- billto.address.street_address_parts.door Door

The following billto.phone_number_parts claim gives back the detalied structure of billto.phone_number claim:

- billto.phone_number_parts.country_code Phone number country code part
- billto.phone_number_parts.number Phone number part

2.4 Payment information

- pcard.holder_name Card holder's name
- pcard.type Card type
- pcard.number Card number
- pcard.verification CVV number
- pcard.expire_month Expiry date - month
- pcard.expire_year Expiry date – year
- pcard.formatted Formatted string of values above

3 List of scopes

The following scopes are defined. As of the writing of this document the scopes defined here are final, but the list may be expanded.

The `scope` parameter of the Authorization endpoint is mandatory, because the scope `openid` is needed to accept the request as a valid OIDC call. So at the very least, this is needed as part of the auth URL:
`scope=openid`

The actual scopes that request user claims are:

- `profile`
- `email`
- `address`
- `phone`
- `billto`
- `pcard`

Each of these will map to a group of user claim, meaning that when you add one to the `scope` parameter, the mapped user claims will get requested. The mapping is:

`profile`:

- prefix
- given_name
- family_name
- middle_name
- name

- gender
- birthdate
- website
- nickname
- preferred_username
- picture
- picture_data
- locale
- zoneinfo

`email`:
- email
- email_verified

`address`:
- address

`phone`:
- phone_number
- phone_number_verified

`billto`:
- billto.name
- billto.company_name
- billto.tax_id
- billto.email
- billto.phone_number
- billto.address

`pcard`:
- pcard.holder_name
- pcard.type
- pcard.number
- pcard.verification
- pcard.expire_month
- pcard.expire_year

4 Usage of the claims parameter

When you specify both the `scope` and `claims` parameter, they complement each other. You can add more claims, or specify what claims will be mandatory.
Let's look at an example value of the `claims` parameter: (URL-decoded and formatted for readability purposes)

```
{
  "userinfo": {
    "email": null,
    "given_name": {"essential": true}
  },
}
```

The parameter value is a two-tier UTF-8-encoded and urlencoded JSON object. The top-level keys define the location where the claims will be returned, and the value of each key defines the claims and its options.

The top-level members of this JSON are:

- `userinfo`
  Claims listed within this object can be retrieved via the UserInfo Endpoint.

The second level of the claims parameter JSON is used to define individual claim requests and its parameters. The keys of this object are the claim names itself, and its values are as follows:

- `null`
  This signifies that the claim is requested as is
- `{"essential": true}`
  This will tell that this claim is mandatory, which means that the end-user will have to fill it out in the profile to complete the authentication. This will also mean that this claim will always be returned in the UserInfo.

Here's the above example again:

```
{
  "userinfo": {
    "email": null,
    "given_name": {"essential": true}
  },
}
```

So according to the specs, this value will require the following claims:

- `email` will be accessible in the UserInfo if given.
- `given_name` will be available in the UserInfo and is mandatory

Note!
That is technically possible to return user-related data in the ID token. If your private key for encrpytion is not well-protected, this solution has got several security risk concern confidentality of the user-related data.
We highly recommend to avoid return user-related data in the ID token.

5 Returned claim structures

The claims `address` and `billto.address` represent JSON structures instead of string values. This means that when you request claims that are prefixed with either of these claims, you will receive it within a JSON structure, with the top key being the parent name of that claim, and the key of the claim itself is without the parent's prefix. Let's see an example:

You request the following in the `claims` parameter:

```
{"userinfo": {"address.locality": null, "billto.address.region": null}}
```

You will receive the values of these in the following manner:

```
"address": {
    "locality": "val"
},
"billto.address": {
    "region": "val"
}
```

Notice how the parent is the key of the JSON structure, and the claim names themselves lose their prefixes.