



# **Technical documentation for implementation of MojED**

*Release 3.1.5*

**CZ.NIC, z. s. p. o.**

**18.06.2024**



# Contents

<b>1</b>	<b>Legal Notice</b>	<b>1</b>
1.1	Limitation of Liability . . . . .	1
1.2	Privacy Protection . . . . .	1
1.3	Applicable Legislation and Competent Jurisdiction . . . . .	1
1.4	Conditions for Using MojelD Logo . . . . .	2
<b>2</b>	<b>Introduction</b>	<b>3</b>
<b>3</b>	<b>Terminology</b>	<b>5</b>
<b>4</b>	<b>Getting started with MojelD</b>	<b>7</b>
4.1	Basics of MojelD . . . . .	7
4.2	MojelD Identity . . . . .	7
4.3	Communication with MojelD . . . . .	8
4.3.1	Communication via OpenID Connect . . . . .	8
4.4	Favicon . . . . .	12
4.4.1	Settings in OpenID Connect . . . . .	12
4.4.2	Settings for SAML . . . . .	14
4.5	Pairing MojelD with NIA . . . . .	14
<b>5</b>	<b>MojelD Support Implementation</b>	<b>15</b>
5.1	Implementation via OpenID Connect (OIDC) . . . . .	15
5.1.1	Overview of Libraries and Modules . . . . .	16
5.1.2	Implementation Process Overview . . . . .	18
5.1.3	Client Registration . . . . .	21
5.1.4	Requesting Login via MojelD . . . . .	24
5.1.5	Initiation . . . . .	24
5.1.6	Requesting Identity Authentication . . . . .	25
5.1.7	Performing Authentication . . . . .	27
5.1.8	Response to Authentication . . . . .	27
5.1.9	Requesting Token . . . . .	28
5.1.10	Requesting Data . . . . .	29
5.1.11	MojelD LITE Library . . . . .	30
5.1.12	Identity verification request with a NIA-paired account . . . . .	32
5.2	Implementation via SAML . . . . .	33
5.2.1	Identity verification request with a NIA-paired account . . . . .	33
5.3	Problems with Implementation . . . . .	33
5.3.1	Differences Between the Protocols . . . . .	34
5.3.2	Transition to a Different Protocol . . . . .	34
5.3.3	Adjusting Communication with MojelD Server . . . . .	34
<b>6</b>	<b>Interface for Creating MojelD Accounts</b>	<b>37</b>
6.1	Requesting Creation of a MojelD Account . . . . .	37
6.2	Checking Data Validity . . . . .	37
6.3	Completing Registration . . . . .	40
<b>7</b>	<b>Logging out of MojelD</b>	<b>43</b>
<b>8</b>	<b>MojelD Test Instance</b>	<b>45</b>
8.1	Test Accounts . . . . .	45
8.2	Mutual Endpoints . . . . .	46

8.3	OpenID Connect . . . . .	46
8.4	SAML . . . . .	47
<b>9</b>	<b>Appendices</b>	<b>49</b>
9.1	Appendix 1 – List of Data to be Handed Over (OpenID Connect) . . . . .	50
9.2	Appendix 3 – List of Data to be Handed Over (SAML) . . . . .	55
9.3	Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz) . . . . .	57
9.4	Appendix 5 – List of Data for Registration . . . . .	60
9.5	Appendix 6 – Examples and Solution of Error Messages . . . . .	64
9.5.1	Error Messages on Test Instance . . . . .	64
9.5.2	Problems Verifying the Return Address . . . . .	64
9.5.3	Problems with Unencrypted Connection . . . . .	67
9.5.4	Selecting Required Logging Method . . . . .	68
9.5.5	Problems with Library for PHP . . . . .	68
9.5.6	Error Messages in JSON (OIDC) . . . . .	68
9.6	Appendix 7 – Correct Implementation Procedure . . . . .	70
<b>10</b>	<b>Record of Changes</b>	<b>73</b>
	<b>Index</b>	<b>77</b>

# Chapter 1

## Legal Notice

### Overview

- [Limitation of Liability](#) (page 1)
- [Privacy Protection](#) (page 1)
- [Applicable Legislation and Competent Jurisdiction](#) (page 1)
- [Conditions for Using MojelD Logo](#) (page 2)

### 1.1 Limitation of Liability

Except in cases of damage caused intentionally or due to serious negligence, or damage to person's basic rights, or to maximal extent allowed by the user's legislation system, the CZ.NIC Association holds no responsibility for any direct or indirect damages resulting from usage (including installation) of MojelD, including but not limited to damage to reputation, damage resulting from interrupted work, loss or damage of data, or any other economical damage (e.g. loss of profits, not reaching expected savings, etc.).

Please keep in mind that the information provided in this documentation does not serve as a warranty, explicit nor implicit, especially as a warranty of suitability for a specific purpose or warranty of usability in other legal system than the legal system of the Czech Republic.

### 1.2 Privacy Protection

MojelD was developed in the Czech Republic and its privacy protection policy is in accordance with the national legislation of the Czech Republic, including opinions of Personal Data Protection Authority. Before using MojelD outside of the Czech Republic make sure that the MojelD data protection policy is in accordance with the legal requirements of the given country.

### 1.3 Applicable Legislation and Competent Jurisdiction

The MojelD implementation documentation (and associated documents) are governed and interpreted in all regards in accordance with Czech legislation. All disputes or claims resulting from or associated to using MojelD (or this documentation), including its interpretation, implementation, invalidity, etc. will be definitively settled by Court of Arbitration at the Economic Chamber of the Czech Republic and Agricultural Chamber of the Czech Republic (hereinafter as "court") pursuant to its Rules of Procedure by one arbitrator elected by this court's chairman.

## 1.4 Conditions for Using MojelD Logo

The CZ.NIC Association is the executor of property copyright rights to figurative mark - MojelD logo and its derived modalities. The CZ.NIC Association hereby allows the usage of the MojelD logo and its associated modalities with regard to implementation, usage and/or promotion of MojelD or promotion of the CZ.NIC Association and its products in any common way logos are used. The right to use MojelD logo and its associated modalities is free of charge, non-exclusive, unlimited in quantity and geography, and limited in time in regard to the usage of MojelD. The user is not required to exercise the right to use the MojelD logo and its associated modalities. The right to use the MojelD logo and its associated modalities cannot be forwarded to a third person without consent from the CZ.NIC Association. The MojelD logo and its associated modalities cannot be abused to damage good reputation of the CZ.NIC Association or used contrary to the interests of the CZ.NIC Association. The MojelD logo and its associated modalities cannot be belittled or used in derogatory manners. The MojelD logo has to be figured as per instructions of the Graphical Manual and used exclusively in such fashion.

# Chapter 2

## Introduction

This document includes a general introduction to the MojelD service. You can also find here examples and other general information that will help you design the implementation of MojelD support in your web application. It will help you get a basic overview of the steps that will have to be taken to implement MojelD support and you will be able to estimate the complexity of the implementation.

MojelD currently offers three authentication protocols that can be used. They are OpenID Connect, OpenID 2.0, and SAML 2.0.

---

**Tip:** If you do not use any of these protocols in your system, we recommend choosing *OpenID Connect*.

It is the newest of the offered protocols and it has some improvements based on the experience from using the other two. Its main advantages are simpler implementation and mobile platforms support.

However, if you already use the OpenID 2.0 or SAML 2.0 protocol in your system, it is logical to use that protocol for integration with MojelD too.

---





# Chapter 3

## Terminology

The following terminology is used in the next chapters regarding the implementation of MojelD:

### Service provider

provider of a web application (or simply an application, because it manages everything automatically without any manual setting) that requires verification of user's *identity* via MojelD

### Full access

MojelD implementation variant at the service provider, more details at <https://www.mojelid.cz/en/provider/options-and-prices/>

### Limited access

MojelD implementation variant at the service provider, more details at <https://www.mojelid.cz/en/provider/options-and-prices/>

### Identity

set of data about the user that are linked to an *identifier* and managed by an OpenID provider

### Identifier

a URL with an http or https`schema that defines and provides certain data in the `:term:`identity <Identity>`, e.g. `http://specs.nic.cz/attr/contact/valid`.

### Realm

the service provider's URL area defining a part of a URL region for which the identity authentication request is valid

### OP

#### OpenID provider

OpenID2 identities provider and maintainer on whose web the authentication is carried out. In case of MojelD, it is the CZ.NIC Association.

### OCP

#### OpenID Connect provider

OpenID Connect identities provider and maintainer on whose web the authentication is carried out. In case of MojelD, it is the CZ.NIC Association.

### Identity name

the name of MojelD *identity* in form of `jmenoidentity.mojelid.cz`, that the user enters in the login form as the identity they want to log in with, e.g. `demo.mojelid.cz`.

### Claimed identifier

identifier derived from identity name under which the identity is available at OpenID provider and from where it is possible to retrieve metadata of this identifier, e.g. `https://demo.mojelid.cz/#UnIqUe`.

### OP endpoint

URL where the OpenID2 provider receives messages. In case of MojelD, it is `https://mojelid.cz/endpoint/`.

### Registration Endpoint

URL where it is possible to register a new service provider according to [OpenID Connect](#)

Dynamic Client Registration<sup>1</sup> specification.

**Client ID**

unique identifier of a service that uses OpenID Connect. It is assigned on registration and used during all the communication via OpenID Connect.

**Client Secret**

password that certifies the service provider's authenticity in regard to his Client ID. This password can be changed using Registration Access Token.

**Registration Access Token**

token used for authorization of any change of data about the service, e.g. Client Secret

**Authorization Endpoint**

a URL to which service providers redirect users for login

**ID Token**

contains a confirmation of a successful identity authentication of a user whose data is contained within the ID Token

**Access Token**

a token used to authenticate a UserInfo Endpoint request

**UserInfo Endpoint**

a URL where it is possible to get detailed data of a user if they are not contained in the ID Token

**Token Endpoint**

a URL where it is possible to get the Access Token, or the Refresh Token, in case they have not been received directly in the response to authentication.

**Refresh Token**

a token that can be used to receive data from the UserInfo Endpoint even without the user's presence.

---

<sup>1</sup> [https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html)

# Chapter 4

## Getting started with MojelD

This chapter is an introduction to basic principles of the MojelD service, the forms of MojelD identities and the communication process via supported protocols.

### 4.1 Basics of MojelD

MojelD is a service that allows its users to create and centrally manage their internet identity (a set of personal data, e.g. first name, surname, e-mail address, phone number, etc. together with login methods). Users can then use this identity to log into various external web applications (applications of different service providers than the identity provider) and they do not have to create individual accounts and repeatedly fill in their basic information and use different usernames and passwords.

The mojeliD service is a specific implementation of the OpenID 2.0 and OpenID Connect 1.0 standard for decentralized management of internet identities which define the ways to verify these centrally managed identities and the forms of their identifiers.

MojelD account can be paired with National Point for Identification and Authentication (NIA) to verify user's identity and gain access to public administration services. For more information, see chapter *Pairing MojelD with NIA* (page 14).

MojelD is specific for the Czech internet environment and offers the service providers additional advantages over the standard OpenID, e.g. extended set of personal data in the identities and their transferring, or more login methods with the possibility to require certain level of authentication.

### 4.2 MojelD Identity

When creating an identity, users have to choose a name of their identity which uniquely determines each MojelD identity and which is always in the form of `identityname.mojelid.cz` (alphanumeric characters), e.g. `demo.mojelid.cz`.

The users then use this name to log into pages of service providers.

MojelD Identity consists of:

- Information the user includes in their identity (common personal data, such as name, address, phone number, nickname, etc.)
- Information about the user provided by the MojelD service provider, especially information about the physical identity verification (user's personal data verification, or the information about whether the person is older than 18).

---

**Tip:** Specific lists of information that can be transferred from the MojelD identity using the individual protocols can be found in *udaje-openid*, *Appendix 1 – List of Data to be Handed Over (OpenID Connect)* (page 50) and *Appendix 3 – List of Data to be Handed Over (SAML)* (page 55).

---

## 4.3 Communication with MojelD

This section generally describes communication processes that take place when a moje ID user logs in to a service that supports a certain protocol.

### 4.3.1 Communication via OpenID Connect

The process of logging in using MojelD has various variants (based on different schemas) that consist of several steps. As you implement MojelD, you can choose the schema(s) you prefer.

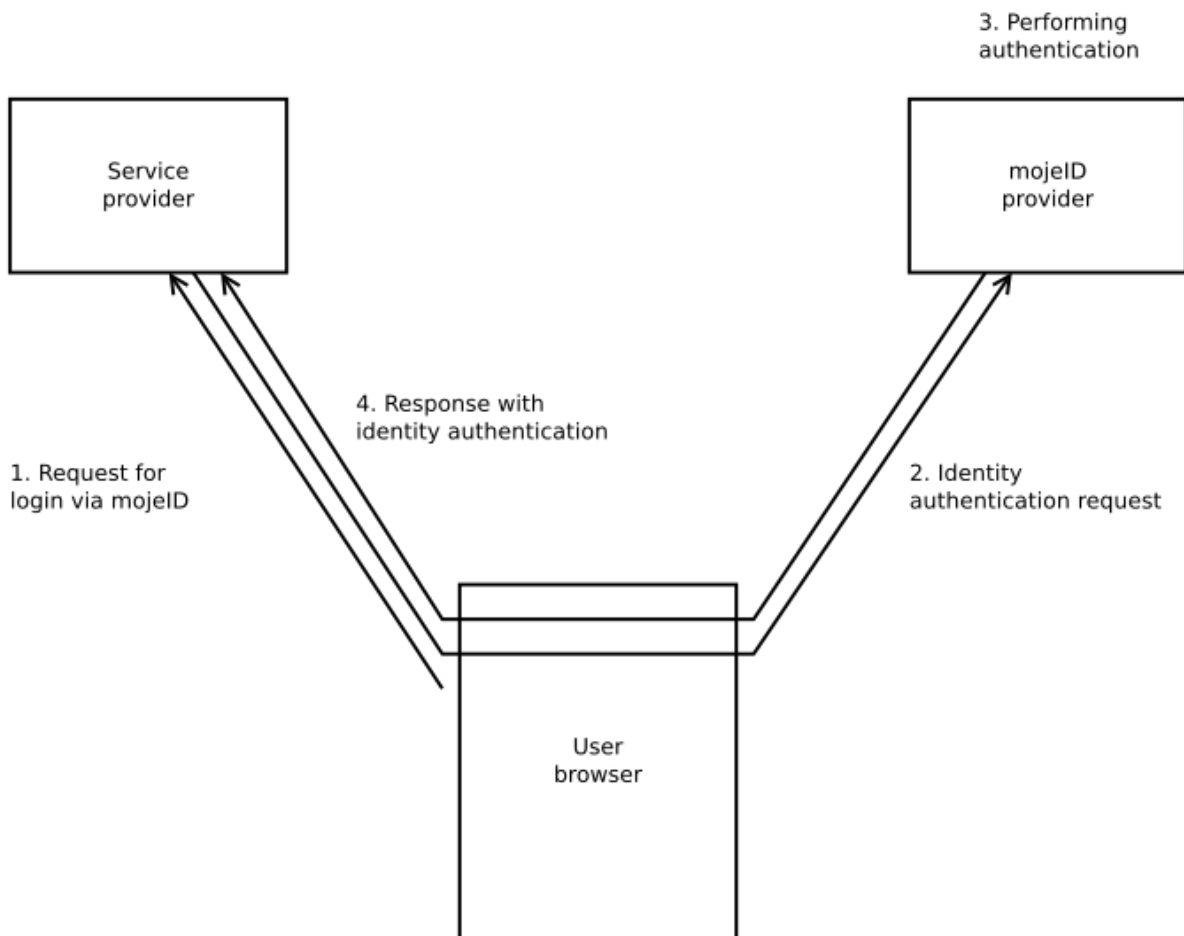
The first steps are the same for all the schemas:

0. **Client's registration** – You have to register your client on MojelD servers before you can use the OpenID Connect protocol.
1. **Requesting login using MojelD** – The user clicks the Log in via MojelD button.
2. **Requesting identity authentication** – The service provider creates an identity authentication request and sends it (indirectly by redirecting the user's browser) to the OpenID Connect provider's endpoint (*Authorization Endpoint*) where the user authenticates.
3. **Performing authentication** – The user logs in at the MojelD login page using one of the login methods to verify their identity. At this moment, we support login with password, digital certificate, one-time password, or security key (FIDO 2).

The next steps depend on the chosen schema.

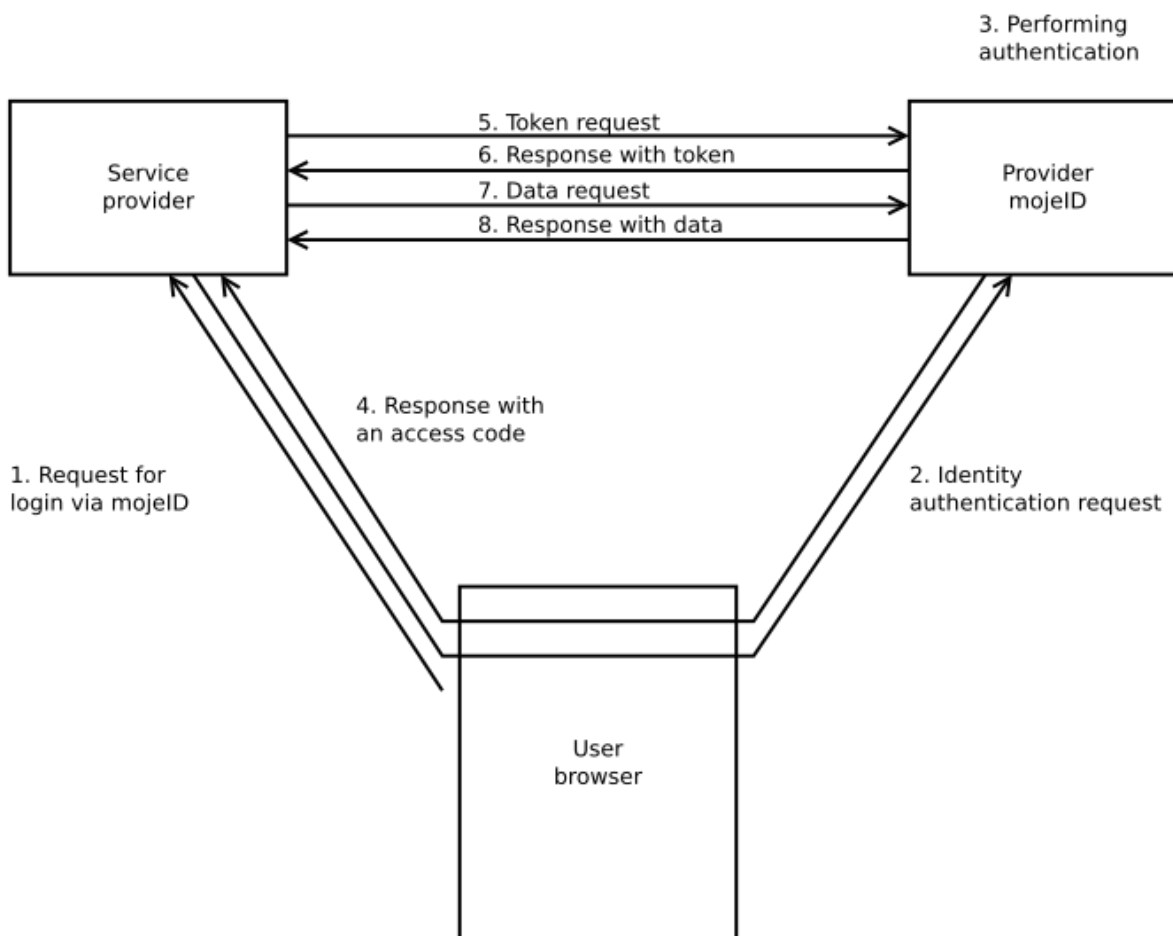
- [Implicit schema](#) (page 9)
- [Access code](#) (page 10)
- [Hybrid schema](#) (page 11)
- [Schema selection](#) (page 12)

### Implicit schema



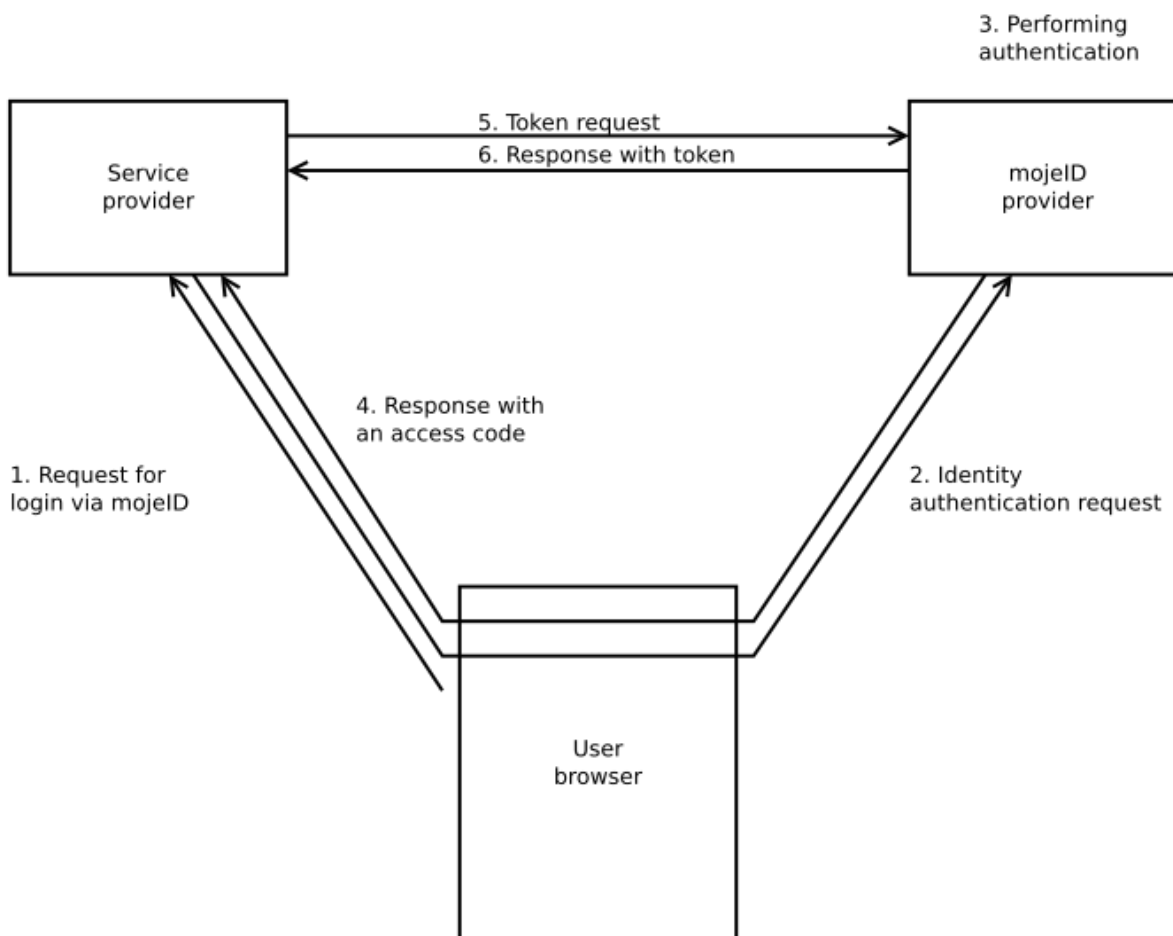
- 4. Response with the identity authentication outcome** – After the login and confirmation, the user is redirected back to the service provider’s website and via their browser sends the response from MojelD servers with the user identifier and ID token. If the service provider requests it during the identity authentication process, the ID token will include data about the user.

## Access code



4. **Response with an access code** – After the login and confirmation, the user is redirected back to the service provider’s website and via their browser sends the response from MojelD servers with an access code.
5. **Token request** – The service provider creates a token request using the access code they just received and sends it to the Token Endpoint.
6. **Response with a token** – The service provider receives a response with access token and token ID.
7. **Data request** – The service provider creates a user data request using the access token they received and sends it to UserInfo Endpoint.
8. **Response with data** – The service provider receives a response with user’s data.

## Hybrid schema



4. **Response with an access code** – After the login and confirmation, the user is redirected back to the service provider’s website and via their browser sends the response from MojelD servers with an access code.
5. **Token request** – The service provider creates a token request using the access code they just received and sends it to the Token Endpoint.
6. **Response with a token** – The service provider receives a response with an access token and a token ID that contains the user’s data.

### Schema selection

For web services that run only in browser (“without server”, e.g. JavaScript), it is best to use *Implicit Schema*.

For server services, it is better to use the *Access Code* schema which is more secure.

The following table provides an overview of the basic characteristics of the individual schemas and it helps with the selection of an appropriate login schema.

Characteristic	Implicit Schema	Access Code	Hybrid Schema
All the tokens are returned from the Authorization Endpoint	yes	no	no
All the tokens are returned from the Token Endpoint	no	yes	no
Tokens are not visible in User Agent	no	yes	no
The client can use authentication	no	yes	yes
It is possible to get a Refresh token	no	yes	yes
Communication within a single request	yes	no	no
Most of the communication is server-to-server	no	yes	various

## 4.4 Favicon

A favicon is a graphical element (icon) associated with a certain website or, in case of MojelD, a service. Web browsers can display favicons as a visual symbol of a website’s identity in address bar, bookmarks or Favourites.

MojelD displays a favicon in the MojelD login form next to the name of the service the MojelD user is logging into.

The use of the favicon differs based on the protocol.

### 4.4.1 Settings in OpenID Connect

You need to upload the favicon file to your website and set its address as metadata (`logo_uri`) within your client’s registration (see *Client Registration* (page 21)).

If the icon is found at the defined URI, it is displayed in the MojelD form, **no matter** the type of access (*full/partial*) služby k MojelD.



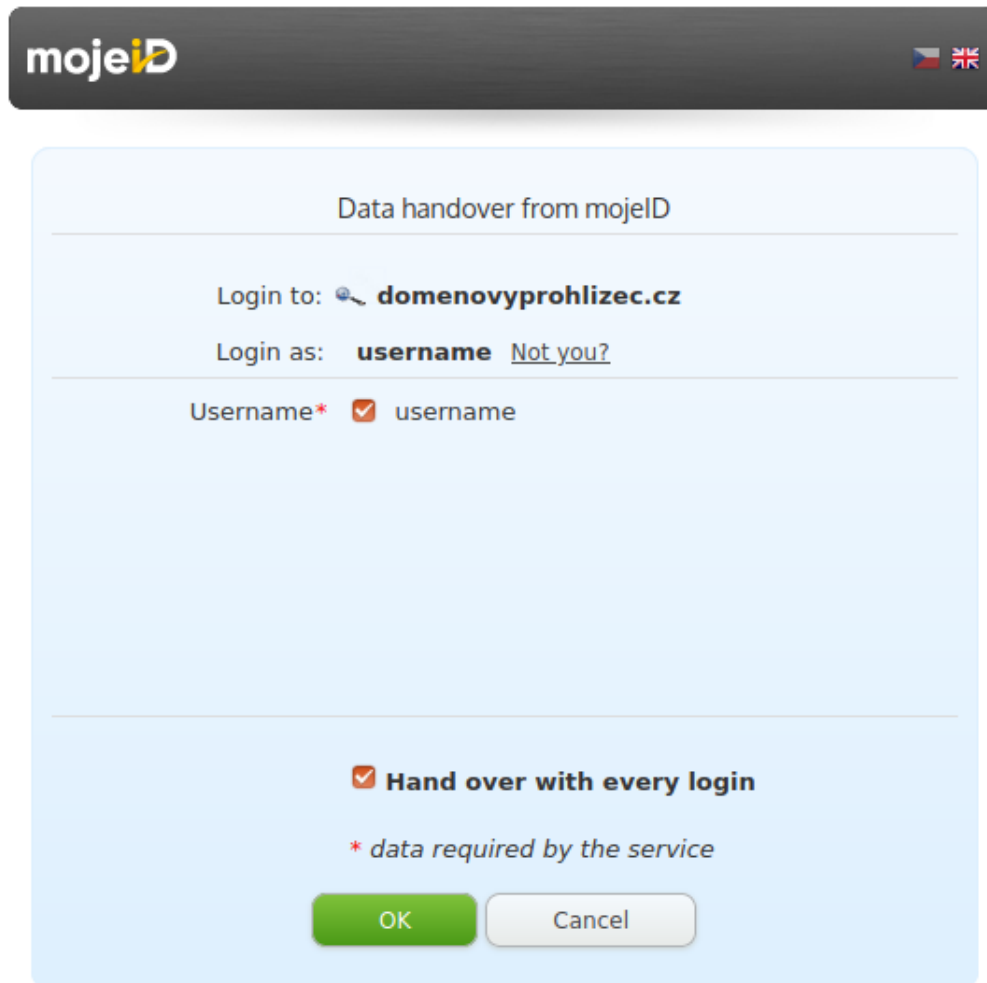


Fig. 1: Favicon display example

### 4.4.2 Settings for SAML

You need to explicitly upload the favicon file to our system.

The favicon is downloaded either automatically (once a week), or you can provide it directly to CZ.NIC (e.g. by e-mail to our support) and we will upload it manually. With automatic downloading, the algorithm searches for the favicon on the provider's *realm* based on the [W3C favicon standard<sup>2</sup>](#), section *Method 1*.

Favicons cannot be larger than 10 kB. The supported formats are ICO and PNG.

Displaying a favicon for services communicating via this protocol is possible only when the service has *full access*.

### 4.5 Pairing MojelD with NIA

MojelD account can be paired with National Point for Identification and Authentication (NIA). Pairing the account verifies the users' identity and the user gains access to public administration services. Only a natural person can be paired. If the `Organization` field is filled, pairing with NIA is not possible.

Transferred data verified by NIA: `First name`, `Surname`, `Home address`, `Date of birth`. Data verified this way cannot be changed in the profile, they are updated automatically from citizen registry. If the user wants to change the locked data, he has to cancel the NIA pairing, also removing the ability to access public administration services. Subsequent data change will also remove identity verification.

MojelD supports two levels of assurance according to eIDAS: "substantial" and "high". Service provider can request login with such an authenticated account using SAML or OIDC protocols only.

More information on how to request such a login can be found in specific protocols:

- OIDC: [Identity verification request with a NIA-paired account](#) (page 32)
- SAML: [Identity verification request with a NIA-paired account](#) (page 33)

---

<sup>2</sup> <http://www.w3.org/2005/10/howto-favicon>

# Chapter 5

## MojelD Support Implementation

This chapter takes you through the details of the individual phases of the communication process that need to be taken into account during the implementation of the support of the protocol. It also describes the prerequisites that need to be met to successfully implement the support.

---

**Important:** Due to security reasons, MojelD does not permit displaying of the login page within frames (`<iframe>`).

---

### 5.1 Implementation via OpenID Connect (OIDC)

This section will introduce you to the technical aspects of the implementation of MojelD into web applications via the OpenID Connect protocol.

We recommend to study this text in order to properly understand the principles and processes of MojelD / OpenID Connect. Most of the things described here can be solved by using available libraries for the implementation of OpenID Connect that we recommend to use.

The *Implementation Process Overview* (page 18) section will take you through the implementation process step by step. Other sections describe the individual steps in more details.

The official specification of the OpenID Connect protocol can be found at [https://openid.net/specs/openid-connect-core-1\\_0.html](https://openid.net/specs/openid-connect-core-1_0.html).

MojelD server publishes basic information about OIDC configuration at <https://mojeid.cz/.well-known/openid-configuration/>.

You can test your implementation using the *MojelD Test Instance* (page 45).

The list of data that can be transferred by the protocol (including their identifiers) is available in the *Appendix 1 – List of Data to be Handed Over (OpenID Connect)* (page 50).

Examples and solutions of error messages can be found in the *Appendix 6 – Examples and Solution of Error Messages* (page 64).

---

**Note:** All the examples of source code listed below illustrate implementation in Python using the `pyoidc` library.

---

### 5.1.1 Overview of Libraries and Modules

The official OpenID Foundation website offers a list of certified OIDC protocol implementations in several programming languages (see [Certified OpenID Connect Implementations](#)<sup>3</sup>). The relevant part for you is implementations for *Relying Party* that corresponds to the service you provide.

For the use in mobile apps, it is best to use libraries for native apps:

- For Android e.g. <http://openid.github.io/AppAuth-Android/>,
- For iOS e.g. <http://openid.github.io/AppAuth-iOS/>.

You can also use modules for the most popular platforms:

- WordPress: [OpenID Connect Generic Client \(daggerheart\)](#)<sup>4</sup>
- Drupal: [OpenID Connect module](#)<sup>5</sup>
- Magento: [OpenID Connect Single Sign-On \(SSO\) Magento Extension By Gluu](#)<sup>6</sup>
- OpenCart: [OpenCart OpenID Connect Single Sign-On \(SSO\) Extension By Gluu](#)<sup>7</sup>
- Moodle: [OpenID Connect Authentication Plugin](#)<sup>8</sup>
- Django: [OIDC Django Packages](#)<sup>9</sup>
- [Login to MojelD via PHP client](#) (page 16)

If you know another one that should be mentioned here, we will be glad to hear from you ([techsupport@mojeid.cz](mailto:techsupport@mojeid.cz)).

### MojelD login via PHP client

This manual contains the procedure for installing the plugin for logging into MojelD via PHP client and an example of how to use it.

### Prerequisites

Before you can proceed, you need to do the following:

- Install [Composer](#)<sup>10</sup>
- Install [Docker Engine](#)<sup>11</sup>
- Download the `php-mojeid-oidc` plugin from our [public GitLab](#)<sup>12</sup>

---

<sup>3</sup> <https://openid.net/developers/certified/>

<sup>4</sup> <https://wordpress.org/plugins/daggerhart-openid-connect-generic/>

<sup>5</sup> [https://www.drupal.org/project/openid\\_connect](https://www.drupal.org/project/openid_connect)

<sup>6</sup> <https://github.com/GluuFederation/magento-oxd-extension>

<sup>7</sup> [https://www.opencart.com/index.php?route=marketplace/extension/info&extension\\_id=27180](https://www.opencart.com/index.php?route=marketplace/extension/info&extension_id=27180)

<sup>8</sup> [https://moodle.org/plugins/auth\\_oidc](https://moodle.org/plugins/auth_oidc)

<sup>9</sup> <https://djangopackages.org/grids/g/oidc/>

<sup>10</sup> <https://getcomposer.org/>

<sup>11</sup> <https://docs.docker.com/engine/install/>

<sup>12</sup> <https://gitlab.nic.cz/mojeid/plugins>

## Installation

1. In the `php-mojeid-oidc` plugin folder run the following commands:

```
cd php
composer install

# create a config file for a specific service
cp config.{template,local}.php

# start a web server
sudo docker compose -f ../docker/docker-compose.yml up
```

2. Do the [manual MojID client registration](#)<sup>13</sup>.
  - In the *list of URIs* fill out URI, that your internet browser uses to access PHP application (folder `php` from this example). If using the provided docker solution on your own computer you can fill out `https://localhost:8443/`.
    - You can find the address the web server uses from the `OpenIDConnectClient::getRedirectURL()` method.
    - If it does not match what you need, set the correct address using the `OpenIDConnectClient::setRedirectURL()` method.
3. In the `config.local.php` file fill out the requested data:
  - `OPEN_ID_PROVIDER_URL` is the base URL of the service to which you want to connect to
  - `OPEN_ID_CLIENT_ID` is the *Client ID* from the page [https://mojeid.regtest.nic.cz/consumer\\_admin/](https://mojeid.regtest.nic.cz/consumer_admin/)
  - `OPEN_ID_CLIENT_SECRET` is the *Client secret* from the page with the details of the service
    - on the page above go to the *Update* link on the corresponding line

## Application

1. Go to the example web page (<https://localhost:8443/>).
2. After potential self-signed certificate confirmation you will be redirected to the MojID login page.
3. During the first login, you will be asked to agree to a data handover.
4. Once agreed you will be redirected back to your application page, where first name of filled user will be displayed *uvidíte křestní jméno zadaného uživatele* (if you have given the appropriate consent).

---

<sup>13</sup> <https://www.mojeid.cz/documentation/html/ImplementacePodporyMojeid/OpenidConnect/Registrace/index.html>

**Important:** We continue to test these modules. We will be glad if you share your experience with them.

---

### 5.1.2 Implementation Process Overview

This overview includes organizational and technical steps you have to take to implement logging in to your service via MojelD using the OpenID Connect protocol. The individual steps are brief and say what to do, while the link targets provide more details on how to do that, or they contain additional information. The overview can serve as a *checklist*.

#### Preparing the test environment

1. [Register your service](#) (page 21) (client) at the [test Registration Endpoint](#) (page 46) – this way you will get test metadata of your service (*Client ID*, *Client Secret*) and an opportunity to set up certain parameters of the communication.

---

**Note:** In case of the *Automatic Registration*, the *Client Secret*'s validity ends after a certain time period. If you decide to opt for *Automatic Registration*, it is important to set up registration renewal.

---

2. Send the service's test metadata (*Client ID*) to support ([techsupport@mojeid.cz](mailto:techsupport@mojeid.cz)). The support sets up accesses.
3. [Create and set up MojelD test accounts](#) (page 45).

#### Implementation and debugging

You will need: text editor, browser, access to hosting, [OIDC specifications](#)<sup>14</sup>

You might find [our recommendations for debug tools](#) (page 34). useful for implementation debugging. During the debugging, you might come across various error messages. [Appendix 6 – Examples and Solution of Error Messages](#) (page 64) might help you with them.

1. [Add MojelD button and links](#) (page 24) to the (template/sites of the) service the user will use to request login. Follow [correct implementation procedure](#) (page 70)!
2. [Get test OIDC provider configuration](#) (page 24) (webfinger).
3. Library configuration – enter test *Client ID* and *Client Secret*, or also test endpoints, if the library cannot retrieve this information automatically from the OIDC provider's configuration.
4. [Create and send an authentication request](#) (page 25) to the [Authorization Endpoint](#) (page 46).

---

<sup>14</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html](https://openid.net/specs/openid-connect-core-1_0.html)

---

**Note:** The request should also include the information about the chosen *authentication schema* `</SeznameniSMojeid/ProcesKomunikacePresMojed/OpenIDConnect/index>`. The following steps correspond to the *Access Code* schema.

---

5. Process the *authentication response* (page 27) at the return address stated in the request which receives an *access code* (`code`).
6. *Create and send a token request* (page 28) to the *Token Endpoint* (page 46). You will use the received *access code* in the request.
7. Process the response from which you get an *Access Token* (`access_token`) and an *ID Token* (`id_token`, [What does ID Token contain?](#)<sup>15</sup>), whose validity has to be verified by the implementation (see [ID Token Validation](#)<sup>16</sup>).
8. If the *ID Token* is valid, *create and send a user data request* (page 29) to *UserInfo Endpoint* (page 46). Use the received *access code* in the request.
9. Process the response with the user's data according to the needs of your service.

### Implementation verification

If you want to operate the service with a full access, we have to perform user test of your implementation before your service transitions to production environment.

1. When you finish debugging your implementation, send a notification to the support team ([techsupport@mojeid.cz](mailto:techsupport@mojeid.cz)) that your implementation is ready for user test and attach the address of your service's test instance.
2. When we finish debugging the last details together, your implementation will be ready for the transition to the production environment.

---

<sup>15</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#CodeIDToken](https://openid.net/specs/openid-connect-core-1_0.html#CodeIDToken)

<sup>16</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#ImplicitIDTokenValidation](https://openid.net/specs/openid-connect-core-1_0.html#ImplicitIDTokenValidation)

### Transition to the production environment

1. To get the full access, you first need to sign a contract.
2. *Register your service* (page 21) (client) at the *production Registration Endpointu* (page 46) – this way you will get production metadata of your service and set up certain parameters of the communication.
3. Send the service's production metadata (Client ID) to the support team ([techsupport@mojeid.cz](mailto:techsupport@mojeid.cz)), also in case of a partial access. The support team will add the service into the catalog.
4. *Get a production OIDC provider configuration* (page 24) (webfinger).
5. Reconfigure the implementation with production metadata, or also endpoints.

**That is all.**



### 5.1.3 Client Registration

To communicate with MojED via OpenID Connect, it is necessary to register a client (service) at the MojED server. It is possible to use either manual, or automatic registration. [Automatic registration](#) (page 21) is suitable for dynamically created clients (JS, mobile devices) and [manual registration](#) (page 21) is suitable for server clients.

#### Manual registration

The manual registration can be done at [https://mojeid.cz/consumer\\_admin/](https://mojeid.cz/consumer_admin/). In case of a MojED test instance, at [https://mojeid.regtest.nic.cz/consumer\\_admin/](https://mojeid.regtest.nic.cz/consumer_admin/). You can then edit and delete the managed clients at the same address. The clients created this way have the validity period set to indefinite. Specifications of individual items can be found in the OpenID Connect protocol document ([https://openid.net/specs/openid-connect-registration-1\\_0.html#ClientMetadata](https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata)).

An example of manual registration of a client in MojED test instance:

1. In any account that you create in the *MojED test instance* (page 45), go to [https://mojeid.regtest.nic.cz/consumer\\_admin/](https://mojeid.regtest.nic.cz/consumer_admin/) after login.
2. Go to the `New service setup` link. Fill in the required fields `Client's name`, `List of URIs` and click `Save`.
  - A record with the client's ID is created in the list of managed services.
3. To get `Client secret / Tajemství klienta` go to the `Update` link in the newly created service.
  - A page where you can edit the setup is displayed – `Client secret` is in the last row of the displayed form.

#### Automatic Registration

More details can be found in the OpenID Connect protocol document ([https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html)). All the necessary settings should be done by the used library. Registration created this way will expire after 24 hours but it can be renewed (see [Registration Change](#) (page 24)).

**Caution:** automatic (dynamic) registration cannot be used for *Full access*.

An example of registering a client using the library:

```
from oic.oic.consumer import Consumer

client = Consumer(SessionDB(URL), OIC_CONFIG, client_config=OIC_CLIENT_CONFIG)
client.redirect_uris = URL + client.consumer_config['authz_page']
provider_info = client.provider_config(ISSUER)
client.register(provider_info["registration_endpoint"], response_types='code',
↳client_name=MY_CLIENT_NAME)
```

An example of a registration query:

```
POST /oidc/registration HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: mojeid.cz

{
  "application_type": "web",
  "redirect_uris":
    ["https://client.example.org/callback",
     "https://client.example.org/callback2"],
  "client_name": "My Example",
  "logo_uri": "https://client.example.org/logo.png",
  "token_endpoint_auth_method": "client_secret_post"
}
```

An example of the server's response to a registration query:

```
HTTP/1.1 201 Created
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "client_id": "s6BhdRkqt3",
  "client_secret": "ZJYCqe3GGRvdrudKyZSOXhGv_Z45DuKhCUkOgBR1vZk",
  "client_secret_expires_at": 1577858400,
  "registration_access_token": "MY.SECRET.REGISTRATION.ACCESS.TOKEN",
  "registration_client_uri": "https://mojeid.cz/oidc/registration?client_
↪id=s6BhdRkqt3",
  "token_endpoint_auth_method": "client_secret_post",
  "application_type": "web",
  "redirect_uris":
    ["https://client.example.org/callback",
     "https://client.example.org/callback2"],
  "client_name": "My Example",
  "logo_uri": "https://client.example.org/logo.png"
}
```

---

**Note:**

**Registration can be processed and Client ID and Client Secret can be retrieved also without the library;**

you only need to send a POST query via curl.

Example:

```
curl --data '{"redirect_uris": "https://navratova-adresa.cz",
  "client_name": "Název služby"}' https://mojeid.cz/oidc/registration/
```

---

Registration also allows to associate metadata with client registration (see [Client Metadata in specification<sup>17</sup>](#)), so the provider can define for example: service name and icon, specifically the attributes `client_name`, `logo_uri`, or `client_uri`.

### Information about Registration

A part of the MojEID server's response to a completed registration is a URL where it is possible to get current information about registration (configuration endpoint `registration_client_uri`), and an access code (`registration_access_token`). When sending a GET query to this URL, it is necessary to authenticate using an access code. It needs to be included in the header of the Authorization HTTP request.

The server's response has the same format as the response to registration and contains current information about your client on our server.

---

<sup>17</sup> [https://openid.net/specs/openid-connect-registration-1\\_0.html#ClientMetadata](https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata)

### Registration Change

You can edit certain information about the registered client using the abovementioned configuration endpoint. Configuration has to be done using a POST query with `registration_access_token` added into the Authorization header. The request format is the same as with the one for registration and its processing on server is also the same, with the following exceptions:

- It is not possible to change the registered `redirect_uri` and `client_id`.
- The `client_secret` value is ignored. In case the item is included in the request, a new `client_secret` is generated. It is sent in the response to the configuration query.

An example of a configuration query that will ensure generation of a new `client_secret` and a change of `logo_uri` and `policy_uri`.

```
POST /oidc/registration?client_id=MYCLIENTID HTTP/1.1
Accept: application/json
Host: mojeid.cz
Authorization: Bearer MY.SECRET.REGISTRATION.ACCESS.TOKEN

{
  "client_secret": null,
  "logo_uri": "https://client.example.org/another-logo.png",
  "policy_uri": "https://client.example.org/policy-page"
}
```

The server's response to the configuration query is the same as the response to the registration query and contains current information about your client on our server.

### 5.1.4 Requesting Login via MojelD

The process of identity authentication starts by the user submitting a login request via MojelD at your website. To ensure maximal user friendliness, you can just use a "Login with MojelD" button, see file *MojelD graphic elements* on the [Getting started](#)<sup>18</sup> page. The username is entered later at the MojelD server.

Logging in to MojelD using a button is the only recommended and correct method.

### 5.1.5 Initiation

To be able to send an identity authentication request, your library needs to know either the user's identifier, or the OCP endpoint.

Your application will use the identifier and endpoint to send a WebFinger query to retrieve details about the OpenID Connect provider. The response to this query includes (among other things):

- **Authorization Endpoint** – this is always `https://mojeid.cz/oidc/authorization/` and this address is used for identity authentication requests.
- **Token Endpoint** – this is always `https://mojeid.cz/oidc/token/` and this address is used for token requests.

---

<sup>18</sup> <https://www.mojeid.cz/en/provider/getting-started/>

- **UserInfo Endpoint** – this is always `https://mojeid.cz/oidc/userinfo/` and this address is used for user data requests.

An example of query for a specific user:

```
GET /oidc/.well-known/webfinger?resource=acct%3A%40mojeid.cz&rel=http%3A%2F%2Fopenid.net%2Fspecs%2Fconnect%2F1.0%2Fissuer HTTP/1.1
Host: mojeid.cz
```

An example of the server's response:

```
HTTP/1.1 200 OK
Content-Type: application/jrd+json

{
  "subject": "acct:joe@mojeid.cz",
  "links": [
    { "rel": "http://openid.net/specs/connect/1.0/issuer",
      "href": "https://mojeid.cz/oidc/" }
  ]
}
```

### 5.1.6 Requesting Identity Authentication

Once you know the OCP endpoint, your application sends an identity authentication request using the user's browser redirection. The request includes special parameters for its realization. Correct use of these parameters is done by the OpenID Connect library used for implementation.

Identity authentication request usually includes the following parameters:

- **Return address (URL) of the application** – The address to which the user returns after logging in from the OpenID Connect provider's website and where the outcome of the login is processed.
- **Required groups of data from MojEID** – An identity authentication request has to contain at least *openid* as a required group of data.
- **Required data from MojEID** – An identity authentication request can also include a list of individual data from the MojEID identity which your application requires and which are handed over to your application with the user's consent after a successful login. For each piece of data, its identifier needs to be presented. The data and its identifiers are listed in [Appendix 1 – List of Data to be Handed Over \(OpenID Connect\)](#) (page 50). This list has a JSON format specified in the [OpenID Connect documentation](#)<sup>19</sup>. Any item can be marked as required using an expression `"essential": true`.

Examples of items that can be included in the identity authentication request are listed in the following table:

<sup>19</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#ClaimsParameter](https://openid.net/specs/openid-connect-core-1_0.html#ClaimsParameter)

Parameter (key)	Description and value
scope	List of required groups of data <i>openid address</i>
response_type	Determining the required authentication schema <i>id_token</i>
client_id	Unique service provider's identifier <i>test_clienti</i>
redirect_uri	Return address from MojID. <i>http://www.poskytovatel-example.cz/</i>
claims	More detailed specification of the required data. <pre>{"userinfo":   {"name": null,    "nickname": {"essential": true}} }</pre>

Example of an authentication request:

```
sid, location = client.begin(path=URL, scope=SCOPE)
HttpResponseRedirect(location)
```

Example of an authentication request query:

Listing 1: Example of requesting data via “scope” (group of data)

```
GET /oidc/authorization/?response_type=code&scope=openid%20profile%20email&
↪client_id=s6BhdRkqt3&state=af0ifjsldkj&redirect_uri=https%3A%2F%2Fclient.
↪example.org%2Fcb HTTP/1.1
Host: mojeid.cz
```

Listing 2: Example of requesting data via “claims” (individual data)

```
GET /oidc/authorization/?state=950ba54cb302a7c6a814f22a4e5c5445&redirect_
↪uri=https%3A%2F%2Fmojeid.cz%3A8000%2Fconsumer%2Foic%2Ffinish%2F&response_
↪type=code&client_id=8ol68PATaSpA&scope=openid&claims=%7B%22userinfo%22%3A+%7B
↪%22name%22%3A+null%2C+%22nickname%22%3A+%7B%22essential%22%3A+true%7D%7D%7D&
↪ui_locales=off HTTP/1.1
Host: mojeid.cz
```

The response from the server comes only after the authentication is performed. Example of the response can be found in the [Response to Authentication](#) (page 27) section.

### 5.1.7 Performing Authentication

When a user comes to the MojelD server with a identity verification request, they see a login page where the login takes place.

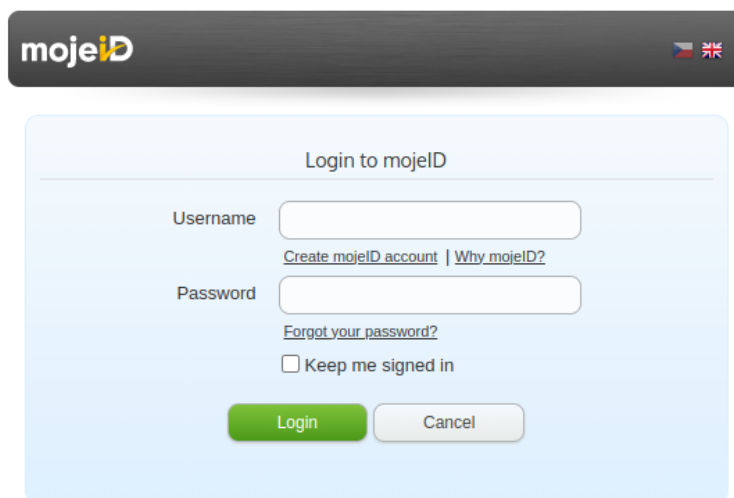


Fig. 1: MojelD login page

This authentication is performed by the MojelD servers. Within this authentication, we will try to perform as many tasks specified by the parameters in the identity authentication request as possible. The whole process takes place exclusively within the MojelD systems and requires no activity from your side.

### 5.1.8 Response to Authentication

When a user completes the authentication process, you will receive a response with its result from the MojelD servers. The structure and contents of this response differs based on the selected communication schema (see [Communication via OpenID Connect](#) (page 8)).

In case of communication via the [Implicit schema](#) (page 9), the response includes the user's identifier and ID Token which can contain data about the user.

In case of communication via [Access code](#) (page 10) or [Hybrid schema](#) (page 11), the response contains an access code that needs to be used in the next step of the authentication process.

An example of processing the response:

```
aresp, _, _ = client.parse_authz(request.GET.urlencode())
```

An example of the server's response:

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?code=Splxl0BeZQQYbYS6WxSbIA&
↳state=af0ifjsldkj
```

### 5.1.9 Requesting Token

If you received an access code in the previous step of authentication, you have to replace it with a valid token at the Token Endpoint.

In case of communication via the *Hybrid schema* (page 11), the response includes an access token and ID Token which can contain data about the user. In this case, the authentication and data transfer process is complete.

In case of communication via an *Access code* (page 10), the response again includes a token and ID Token, but it does not contain any data about the user. You have to request them in the next step.

An example of communication:

```
POST /oidc/token/ HTTP/1.1
Host: mojeid.cz
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3FOMzpnWDFmQmFOM2JW

grant_type=authorization_code&code=Sp1xl0BeZQQYbYS6WxSbIA&redirect_uri=https%3A
→%2F%2Fclient.example.org%2Fcb
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "access_token": "S1AV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xL0xBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc
    yI6ICJodHRwOi8vc2VydmVyLmV4YW1wbGUuY29tIiwiaWF0IjoiAUMjQ4Mjg5
    NzYxMDAxIiwiaWF0IjoiAicZCaGRSa3FOMyIsCiAibm9uY2UuOiAibi0wUzZ
    fV3pBMk1qIiwiaWF0IjoiAicZCaGRSa3FOMyIsCiAibm9uY2UuOiAibi0wUzZ
    AKfQ.ggW8hZ1EuVLuxNuuIJKX_V8a_OMXzROEHR9R6jgdqr00F4daGU96Sr_P6q
    Jp6IcmD3HP990bi1PRs-cwh3L0-p146waJ8IhehcwL7F09JdijmBqkvPeB2T9CJ
    NqeGpe-gccMg4vfKjkM8FcGvzZUN4_KSP0aAp1t0J1zZwgjxqGBYKHItX7Tpd
    QyHE5lcMiKPXfEIQILVq0pc_E2DzL7emopWoaoZTF_m0_NOYzFC6g6EJb0EoRoS
    K5hoDalrcvRyLSrQAZZKflyuVCyixEoV9GfNQC3_osjzw2PAithfubEEBLuVVk4
    XUVrWOLrLl0nx7RkKU8NXNHq-rvKMzqg"
}
```



### 5.1.10 Requesting Data

In this step you will use the token received in the previous authentication step to get the user's data. The data needs to be retrieved from the UserInfo Endpoint.

The UserInfo Endpoint always returns an attribute `sub` (*subject*), in the response which uniquely identifies the user and should be used to validate the response using an *ID Token*.

The user's data should be processed only in case the response is found valid.

An example of requesting data:

```
state = aresp.to_dict()['state']
resp = client.complete(state)
userinfo = client.get_user_info(state)
```

An example of communication with server:

```
GET /oidc/userinfo/ HTTP/1.1
Host: mojeid.cz
Authorization: Bearer SLAV32hkKG
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "sub": "248289761001",
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@example.com"
}
```

### 5.1.11 MojelD LITE Library

Javascript library **MojelD LITE** (or also MojelD Connect) allows to load data from a MojelD identity to a website on the client's side using the OpenID Connect protocol.

This feature can be used, for example, to simply prefill a web form with data of a user with an active MojelD account.

To enable this feature in your web form, you have to perform at least the following steps:

1. *Insert a link to the library.*

If you want to decrease your dependency on an external website, you can upload this library to your own website. The library can be downloaded [here](https://www.mojelid.cz/public/media/1542958574/150/)<sup>20</sup>. The library depends on a cryptographic library [jsrsasign](https://kjur.github.io/jsrsasign/)<sup>21</sup> which is available (in its newest version) on our website, so you do not have to insert it directly. The code of the script to insert the library has to be inside <HEAD>.

An example of inserting the library:

```
<script type="text/javascript"
  src="https://www.mojelid.cz/public/media/1542958574/150/"
  data-jsrsasign="https://www.mojelid.cz/public/media/1542956522/149/">
</script>
```

2. *Call a function for creating a MojelidConnect object.*

This object represents communication with MojelD server. When calling the creating function, you can *set certain parameters* (page 31), that will affect the data transfer process. The code of the script to call the function has to be inside <HEAD>.

An example of creating the object:

```
<script type="text/javascript"> (function() {
  mojelid = createMojelidConnect( {
    clientName: "Sample form",
    claims: ['phone_number', 'family_name', 'given_name', 'nickname',
            'email', 'address', 'birthdate', 'gender', 'website', 'profile']
  } );
})();</script>
```

3. *Attach calling of requestAuthentication() method to the button that activates the prefilling of the form.*

This method initiates the authentication process and filling the form with the values of the confirmed data.

An example of a code for the button:

```
<button onclick="mojelid.requestAuthentication()">
Prefill using MojeID
</button>
```

---

<sup>20</sup> <https://www.mojelid.cz/public/media/1542958574/150/>

<sup>21</sup> <https://kjur.github.io/jsrsasign/>

### createMojelDConnect(options) function parameters

When calling this function, you can set certain parameters (in dictionary structure) that will affect communication with the MojelD server:

#### clientId

It is possible that the service is already registered in the MojelD server. If yes, this service has a clientId assigned and you can provide it in the parameter. If the clientId parameter is not defined, registration is completed dynamically according to the [OpenID Connect specifications](#)<sup>22</sup> using the address from the regEndpoint parameter. **Caution:** automatic (dynamic) registration cannot be used for [Full access](#).

#### clientName

In case of dynamic registration, it is possible to define the name of the service that is shown to the user upon data transfer approval. If the name is not defined, the service's URL is used.

#### scope

Required transferred data in form of a group of data. The value is a sublist ['openid', 'profile', 'email', 'phone', 'address'], while 'openid' is required. If it is not defined, the value is ['openid'].

#### claims

Required transferred data in form of individual attributes. The value is a list of attributes. A full list of possible attributes is available in the value of claims\_supported from [server's configuration file](#)<sup>23</sup>. An example of a list: ['phone\_number', 'family\_name', 'given\_name', 'nickname', 'email', 'address', 'birthdate', 'gender', 'website', 'profile']

#### attrDict

The library assumes the form items have the same id as the name of the attribute from the claims list. If that is not the case, it is possible to define a map list for the form item id and for the attribute name in this parameter.

#### formCallback

If the map dictionary from attrDict is not sufficient, you can define a name of your own JS function that will take care of filling the form.

#### display

The value is either popup or redirect based on whether the login should be done in a new window or in the existing one. The default value is popup.

#### regEndpoint

Registration endpoint's URL according to the [OpenID Connect protocol specification](#)<sup>24</sup>. The default value is <https://mojeid.cz/oidc/registration/>.

#### authEndpoint

---

<sup>22</sup> [https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html)

<sup>23</sup> <https://mojeid.cz/oidc/.well-known/openid-configuration/>

<sup>24</sup> [https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html)

Authentication endpoint's URL according to the [OpenID Connect protocol specification](#)<sup>25</sup>. The default value is `https://mojeid.cz/oidc/authorization/`.

### Sample form

For easier understanding, you can have a look at and try a [full form sample](#)<sup>26</sup>.

### 5.1.12 Identity verification request with a NIA-paired account

Identity verification request with a NIA-paired MojEID account is requested using `acr_values` parameter. Values for requesting specific level of assurance are summed in the table below.

ACR value	Description
<code>http://eidas.europa.eu/oidc/level-of-assurance/substantial</code>	EIDAS level of assurance "substantial"
<code>http://eidas.europa.eu/oidc/level-of-assurance/high</code>	EIDAS level of assurance "high"

Detailed information about `acr_values` can be found in the OpenID Connect documentation on the following links:

- [ID Token](#)<sup>27</sup>.
- [Authentication Request](#)<sup>28</sup>.
- [Requesting the "acr" Claim](#)<sup>29</sup>.

---

<sup>25</sup> [https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html)

<sup>26</sup> <https://www.mojeid.cz/public/media/1542960671/153/>

<sup>27</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#IDToken](https://openid.net/specs/openid-connect-core-1_0.html#IDToken)

<sup>28</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#AuthRequest](https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest)

<sup>29</sup> [https://openid.net/specs/openid-connect-core-1\\_0.html#acrSemantics](https://openid.net/specs/openid-connect-core-1_0.html#acrSemantics)

## 5.2 Implementation via SAML

SAML is a protocol that historically precedes the newer OpenID protocols. If your system already supports SAML (for example an installation of Shibboleth system or similar), it is also possible to use this protocol to enable MojelD.

SAML 2.0 implementation is based on specifications available at <https://wiki.oasis-open.org/security/FrontPage>

To enable MojelD, you need to send the service’s metadata to [techsupport@mojeid.cz](mailto:techsupport@mojeid.cz), and you might also need to register MojelD metadata listed at <https://mojeid.cz/saml/idp.xml>. The certificate listed in metadata can change, so the metadata need to be updated from time to time. Metadata signature can be verified using the certificate at <https://mojeid.cz/saml/cert>.

Because SAML messages are *base64-encoded* and *deflated*, you can convert them to a readable XML for the debugging purposes (you can use for example <https://www.samltool.com/decode.php>).

The list of data that can be transferred by the protocol (including their identifiers) is available in the [Appendix 3 – List of Data to be Handed Over \(SAML\)](#) (page 55) and [Appendix 4 – List of Data to be Handed Over \(SAML specs.nic.cz\)](#) (page 57).

Examples and solutions of error messages can be found in the [Appendix 6 – Examples and Solution of Error Messages](#) (page 64).

### 5.2.1 Identity verification request with a NIA-paired account

Identity verification request with a NIA-paired MojelD account is requested using `AuthnContextClassRef` (Authentication Context Class Reference) class. Values for requesting specific level of assurance are summed in the table below.

<code>AuthnContextClassRef</code>	Description
<code>http://eidas.europa.eu/LoA/substantial</code>	EIDAS level of assurance “substantial”
<code>http://eidas.europa.eu/LoA/high</code>	EIDAS level of assurance “high”

Usage example:

```
<saml:AuthnContextClassRef xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
  http://eidas.europa.eu/LoA/substantial
</saml:AuthnContextClassRef>
```

## 5.3 Problems with Implementation

This section informs about some possible problems with implementation and offers ways to solve them or avoid them.

### 5.3.1 Differences Between the Protocols

A major difference between the protocols is that each protocol can hand over only certain data from the MojelD identity and the set of data is different for each protocol.

We work on their unification, but at this moment, **it is not possible** to hand over all the identity data using each supported protocol.

The data that are handed over for each protocol are listed in the following appendices:

- [Appendix 1 – List of Data to be Handed Over \(OpenID Connect\)](#) (page 50),
- [Appendix 3 – List of Data to be Handed Over \(SAML\)](#) (page 55),
- [Appendix 4 – List of Data to be Handed Over \(SAML specs.nic.cz\)](#) (page 57).

### 5.3.2 Transition to a Different Protocol

In general, the transition to a different protocol is performed by the user logging in to a service using one of the current login methods and then they login again using the new protocol. This way, the service provider can assign the existing user a new protocol identifier.

#### Transition from OpenID 2.0 protocol to the new OpenID Connect protocol

If you want to transition from the original OpenID 2.0 protocol to the latest OpenID Connect, send an identity authentication request via the OpenID Connect protocol with the scope parameter extended with an openid2 value, and you will receive an OpenID 2.0 identity together with an OpenID Connect identity.

More details about the migration process can be found in these [specifications](#)<sup>30</sup>.

### 5.3.3 Adjusting Communication with MojelD Server

To debug communication issues, we recommend to use developer tools in the internet browser. They enable checking network activities: queries and responses exchanged between the client (your implementation) and the MojelD server. This can help you detect a possible error in the data that are handed over.

---

**Note:** For more complicated issues, when you have to contact out technical support, it is useful to attach a recorded communication log to the description of the issue.

---

In **Firefox**, you can use built-in tools or extensions (e.g. FireBug):

1. The developers tools can be enabled in *Main Menu* → *Web Developer* or by a shortcut `Ctrl+Shift+I`.
2. Then you switch to the *Network* tab (or call this tab directly using the shortcut `Ctrl+Shift+Q`).

In **Chrome**, you can use built-in tools:

1. The developers tools can be enabled in *Main menu* → *More Tools* → *Developer tools* or by a shortcut `Ctrl+Shift+I`.

---

<sup>30</sup> [https://openid.net/specs/openid-connect-migration-1\\_0.html](https://openid.net/specs/openid-connect-migration-1_0.html)

2. Then switch to the *Network* tab.

### Debugging in a Pop-up Window

If you implement the user authentication via MojelD using a new pop-up window, you need to do the following to record the communication:

1. Let the pop-up window generate for the first time.
2. Before sending the request to the MojelD server, right click inside it and open the debug tool by choosing the following item in the menu:
  - Chromium: *Inspect*
  - Firefox: *Inspect Element*
  - FireBug plugin: *Inspect Element in Firebug*
3. Call a pop-up window refresh (e.g. F5 or Ctrl+R).
4. Continue reporting network communication in the debug tool in a normal way.





## Chapter 6

# Interface for Creating MojelD Accounts

This chapter describes the process of registration MojelD accounts via your application.

### 6.1 Requesting Creation of a MojelD Account

The user selects an option of creating a MojelD account within your application. That will generate a HTTPS POST request to a registration server in user's browser at <https://direct.mojelid.cz/registration/direct/>. The parameters of the request contain a username and all the other available data about the user (the list of data for registration include [Appendix 5 – List of Data for Registration](#) (page 60)), plus:

- **the service provider identifier** (*realm*) – a selectable URI with a value based on the type of the communication protocol:
  - in case of OpenID 2.0, it should be the same value as the one used for logging into MojelD,
  - in case of OpenID Connect, it must be the assigned `client_id`,
- **unique transaction identifier** (*registration\_nonce*) – used to match the response to this request.

You also have the possibility to offer the user a transfer of an existing account in the central register by choosing the address <https://mojelid.cz/transfer/endpoint/>. In such case, the data about the user are ignored and the username (= contact identifier) that cannot be changed is used. If the identifier is invalid, it cannot be transferred to MojelD and the user has to contact the current registrator to ask for change.

Then the user is displayed a list of data to be entered into MojelD once the registration is completed. The basic data also shows its value and it can be changed. The user will then consent to the service usage policy within the registration form and it will be verified with CAPTCHA.

### 6.2 Checking Data Validity

After a registration form is submitted, the registration server checks data validity and asks the user to correct any errors. In case the data is valid, the process of registration of a new account is initiated. The registration server saves all the necessary data in this account and adds your identification (service provider identifier, realm). Then, the identification of the user starts with sending verification codes to e-mail and phone number.

The next step is informing your application of a successful registration.

In case of communication via OpenID 2.0, the server uses the URI that identifies your realm and tries to find an XRDS document with at least one `<xrd:Service>` element containing the following elements:

- `<xrd:Type>` with a value of [http://specs.nic.cz/registration/assert\\_url](http://specs.nic.cz/registration/assert_url) and

- `<xrd:URI>` with a URL interface where the information about registration is sent.

During this process, there must not be any redirection and the URL interface must be within the service provider's URI (realm) (see [https://openid.net/specs/openid-authentication-2\\_0.html#realms](https://openid.net/specs/openid-authentication-2_0.html#realms)).

In case of communication via OpenID Connect, the URL for sending information must be entered during the *client registration* (page 21) process using the `assertion_uris` key that contains a list of addresses (encrypted in a JSON) to send the messages to.

Your application directly sends a HTTPS POST message to the interface determined by the URL. The message contains three parameters:

- `registration_nonce` – a unique transaction identifier for matching with the original request,
- MojelD user's identifier based on the used protocol:
  - `claimed_id` – in case of the OpenID 2.0 protocol,
  - `sub` – in case of the OpenID Connect protocol,
- `status` – status with the value `REGISTERED`.

Your application first has to verify this message:

- it has to check if the message was delivered to one of the addresses listed in the *Requesting Creation of a MojelD Account* (page 37) of a MojelD Account section,
- it has to check if the `registration_nonce` transaction was really created,
- it has to check if the client certificate used to create an SSL tunnel is valid and signed by the certificate authority, the CZ.NIC Association. This certificate is available at <https://www.mojelid.cz/en/provider/getting-started/#download> for both production and test environment. The certificate is needed for notification in both the production and test environment.

If you do not use HTTPS and you want to try logging in and creating accounts in the test environment, you do not need this certificate.

If you use HTTPS and you are in the test environment, you need this certificate to send notifications from registration. It is not needed for logging in (only general public data is transferred between MojelD and your server, so it is not necessary to check the "identity" of the requester).

The notifications are sent after registration, partial identification (verified e-mail and phone) and identification (entered PIN3, until 2024 only) to `assert_url` listed in the XRDS document in the realm. This works also in the test environment. If you want your application to be able to receive notifications, you need a realm with HTTPS. When the notification is received, it is necessary to respond with a `'mode:accept\n'` string, where the new lines are marked with `\n`.

---

**Tip:** The client certificate verification can be done by an HTTP server, e.g. Apache with the `SSLVerifyClient` configuration option.

---

If all the requirements are met, your application can match the MojelD identifier with its record of the user during the processing of this message for the purpose of authentication via MojelD.

---

**Note:** If this message cannot be sent securely using HTTPS, the registration continues without sending this message.

---

## 6.3 Completing Registration

Your application sends a response to the message from the [Checking Data Validity](#) (page 37) section in the body of a HTTP response in a form of key-value of the OpenID protocol

- **outcome** (*mode*) – value `accept` or `reject` indicating, whether the user's account was successfully paired,
- **reason for denial** (*reason*) – an optional parameter that includes the reason the pairing was not performed.

If a response in the correct format is not received, the message with the result of the registration will be sent to another address from the [Checking Data Validity](#) (page 37) section, until a response is obtained or until all the addresses are used.

The registration then continues either with a direct request to verify e-mail and phone number and going to the user's profile where they choose a password, or the user is shown an information about the completion of the registration.

If you have activated the [full access](#), your application will also receive information about a change of the status of the user's account. These messages are sent in a similar way as described in the [Checking Data Validity](#) (page 37) section, with two parameters in each message:

- **MojelD user's identifier based on the used protocol:**
  - `claimed_id` – in case of OpenID 2.0,
  - `sub` – in case of OpenID Connect.
- `status` – **account status, one of the following values:**
  - `CONDITIONALLY_IDENTIFIED` – **partially identified (PIN1 and PIN2 entered).**
    - \* Account with verified e-mail and phone number.
    - \* PIN1 and PIN2 entered for accounts up to 2024.
  - `IDENTIFIED` – **identified (PIN1, PIN2 and PIN3 entered<sup>31</sup>).**
    - \* Only for accounts up to 2024.
  - `VALIDATED` – **validated (account with validation flag.)**
    - \* Validated account of a business person or natural person's account connected to the public administration services (NIA).
    - \* For accounts up to 2024 this means entered PIN1, PIN2, PIN3<sup>32</sup> and the validation flag.

If this message cannot be sent or no response is received, the information of the change of state will be sent repeatedly each five minutes for the period of six hours, until your application accepts or refuses it. On the other hand, the message about the completion of the registration is synchronous – it is sent only once.

---

<sup>31</sup> The PIN3 for identification of the MojelD account is optional. Identification can be obtained by pairing the account with NIA or by validation. Therefore, there can be a situation when the user has an identified account with only PIN1 and PIN2 entered.

<sup>32</sup> The PIN3 for validation of the MojelD account is optional. Therefore, there can be a situation when the user has a validated account with only PIN1 and PIN2 entered.

As of July 2022 personal accounts cannot be verified using PIN3. PIN3 verification is available only to accounts with the Organization field filled in.

---

**Important:** Since 2024 PIN1, PIN2 and PIN3 are not used for verification.

---



## Chapter 7

# Logging out of MojelD

It is logical based on the way MojelD works that your service cannot automatically log a user out because it would log them out of other services they are logged into via MojelD too. However, in rare cases, a user might need to be logged out of MojelD too. For example, when they logged in from someone else's device.

Then it is desirable that upon or after logging out of your service, the user is asked if they want to be logged out of MojelD too.

If the user chooses this option, redirect them or provide a link to <https://mojeid.cz/logout/> where they can confirm the logout.

We recommend implementing this option if users often access your service from public devices (e.g. in a library or internet café) and if it is not securely solved, e.g. by deleting data after finishing working with the browser.

However, its implementation is not mandatory.





# Chapter 8

## MojelD Test Instance

It is possible to test your implementation using our MojelD test instance where you can test logging of MojelD users, registering of new accounts and transferring of accounts from the central register.

**Before you start testing**, send the metadata you are going to use for testing to [techsupport@mojeid.cz](mailto:techsupport@mojeid.cz). This metadata differs for each protocol (see information about the individual protocols below).

---

**Important:** Use different metadata than for the production instance!

---

We will grant you access to the test instance and set up a so-called *full access*, for the purpose of testing, so that you can receive all the MojelD account data, including `status`, `valid` and more that are transferred only to the providers with *full access*.

### 8.1 Test Accounts

To test MojelD, we recommend creating three test users with different levels of verification. Use the manual on the main page of [MojelD public test instance](#)<sup>33</sup> to create the accounts. You can fill in any contact and personal information.

- **Partially identified account:**
  - Account with verified e-mail and phone number.
- **Natural person's account connected to public administration services:**
  - To connect test account to public administration services you will need a certified hardware or system security key.
  - Create an account for personal use.
  - Click `Verify identity`, then `Verify differently` and select `Test Profile High`.
  - Choose any test profile and complete the verification.
- **Validated account of a business person / organization:**
  - Create an account for business use.
  - Go to the tab with personal information and click `Validate`.
  - Download the generated PDF document and send it to [techsupport@mojeid.cz](mailto:techsupport@mojeid.cz).
  - We will set the validation flag for this account.

This allows you to test returned values in the `status` parameter for all current account verification types.

---

<sup>33</sup> <https://mojeid.regtest.nic.cz/index.html>

## 8.2 Mutual Endpoints

Part of the interface addresses does not depend on the selected protocol. Those addresses are listed here. However, you will also need addresses of endpoints specific for individual protocols that are listed below.

A test instance with more detailed outputs in case of errors is available at the following addresses:

- Registering a new MojelD account: <https://mojeid.regtest.nic.cz/registration/endpoint/>
- Transferring a contact to MojelD from the domain registry: <https://mojeid.regtest.nic.cz/transfer/endpoint/>

The following addresses will be available to implement MojelD to production environment:

- Registering a new MojelD account: <https://mojeid.cz/registration/endpoint/>
- Transferring a contact to MojelD from the domain registry: <https://mojeid.cz/transfer/endpoint/>

## 8.3 OpenID Connect

### Metadata that need to be sent to support

- `Client_ID` you will use for testing – a combination of 12 characters (lower- and uppercase letters and digits) generated automatically upon the registration of the service

### Specific endpoints for the protocol

- **Addresses of the test endpoints:**

- Registration Endpoint: <https://mojeid.regtest.nic.cz/oidc/registration/>
- Authorization Endpoint: <https://mojeid.regtest.nic.cz/oidc/authorization/>
- Token Endpoint: <https://mojeid.regtest.nic.cz/oidc/token/>
- UserInfo Endpoint: <https://mojeid.regtest.nic.cz/oidc/userinfo/>

A full description of OIDC configuration in JSON: <https://mojeid.regtest.nic.cz/.well-known/openid-configuration/>

- **Addresses of the production endpoints:**

- Registration Endpoint: <https://mojeid.cz/oidc/registration/>
- Authorization Endpoint: <https://mojeid.cz/oidc/authorization/>
- Token Endpoint: <https://mojeid.cz/oidc/token/>
- UserInfo Endpoint: <https://mojeid.cz/oidc/userinfo/>

A full description of OIDC configuration in JSON: <https://mojeid.cz/.well-known/openid-configuration/>

## 8.4 SAML

The metadata of the test instance are available at: <https://mojeid.regtest.nic.cz/saml/idp.xml>

### Metadata that need to be sent to support

- string `entityID` you will use for testing – maximal length 1024 characters, specifications recommend the string to be in a form of [URL](#)<sup>34</sup> and to include a domain name of the provider or the provided service

Example: `https://sluzba.example.cz`

- an XML file with the service metadata (`EntityDescriptor`), that contains the same `entityID`

You can find more details on how to get the file with metadata in this [article about metadata preparation](#)<sup>35</sup>.

### Endpoints specific for the protocol

- test endpoint: `https://mojeid.regtest.nic.cz/saml/`
- production endpoint: `https://mojeid.cz/saml/`

---

<sup>34</sup> <https://en.wikipedia.org/wiki/URL#Syntax>

<sup>35</sup> <https://www.eduid.cz/en/tech/metadata-preparation>



# Chapter 9

## Appendices

### List of Appendices

- *Appendix 1 – List of Data to be Handed Over (OpenID Connect)* (page 50)
- *Appendix 3 – List of Data to be Handed Over (SAML)* (page 55)
- *Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)* (page 57)
- *Appendix 5 – List of Data for Registration* (page 60)
- *Appendix 6 – Examples and Solution of Error Messages* (page 64)
- *Appendix 7 – Correct Implementation Procedure* (page 70)

## 9.1 Appendix 1 – List of Data to be Handed Over (OpenID Connect)

Data	Claim identifier	Data type
<b>OpenID2</b> identifier for migration from an older protocol	openid2_id	<i>SINGLE_OPTIONAL_STRING</i>
<b>Name</b>		
Whole name	name	<i>SINGLE_OPTIONAL_STRING</i>
First name	given_name	<i>SINGLE_OPTIONAL_STRING</i>
Surname	family_name	<i>SINGLE_OPTIONAL_STRING</i>
Nickname	nickname	<i>SINGLE_OPTIONAL_STRING</i>
<b>Email</b>		
Main	email	<i>SINGLE_OPTIONAL_STRING</i>
Flag – email verified	email_verified	<i>SINGLE_OPTIONAL_BOOLEAN</i>
Notify	mojeid_email_notify	<i>SINGLE_OPTIONAL_STRING</i>
Other	mojeid_email_next	<i>SINGLE_OPTIONAL_STRING</i>
<b>Home address</b>		
Full address	mojeid_address_def	<i>OPTIONAL_ADDRESS_STRING</i>
Street	mojeid_address_def_street	<i>SINGLE_OPTIONAL_STRING</i>
Street 2	mojeid_address_def_street2	<i>SINGLE_OPTIONAL_STRING</i>
Street 3	mojeid_address_def_street3	<i>SINGLE_OPTIONAL_STRING</i>
City	mojeid_address_def_city	<i>SINGLE_OPTIONAL_STRING</i>
State	mojeid_address_def_state	<i>SINGLE_OPTIONAL_STRING</i>
ZIP code	mojeid_address_def_postal_code	<i>SINGLE_OPTIONAL_STRING</i>
Country	mojeid_address_def_country	<i>SINGLE_OPTIONAL_STRING</i>
<b>Mailing address</b>		
Full address	address	<i>OPTIONAL_ADDRESS</i>
Street	mojeid_address_mail_street	<i>SINGLE_OPTIONAL_STRING</i>
Street 2	mojeid_address_mail_street2	<i>SINGLE_OPTIONAL_STRING</i>
Street 3	mojeid_address_mail_street3	<i>SINGLE_OPTIONAL_STRING</i>
City	mojeid_address_mail_city	<i>SINGLE_OPTIONAL_STRING</i>
State	mojeid_address_mail_state	<i>SINGLE_OPTIONAL_STRING</i>
ZIP code	mojeid_address_mail_postal_code	<i>SINGLE_OPTIONAL_STRING</i>
Country	mojeid_address_mail_country	<i>SINGLE_OPTIONAL_STRING</i>

continues on next page

Table 1 – continued from previous page

Data	Claim identifier	Data type
Flag – address verified <i>Only for full access</i> ("true"/"false") As of July 2022, the flag cannot be obtained for new personal accounts as they cannot be verified using PIN3.	mojeid_address_mail_verified	<i>SINGLE_OPTIONAL_BOOLEAN</i>
<b>Billing address</b>		
Full address	mojeid_address_bill	<i>OPTIONAL_ADDRESS_STRING</i>
Street	mojeid_address_bill_street	<i>SINGLE_OPTIONAL_STRING</i>
Street 2	mojeid_address_bill_street2	<i>SINGLE_OPTIONAL_STRING</i>
Street 3	mojeid_address_bill_street3	<i>SINGLE_OPTIONAL_STRING</i>
City	mojeid_address_bill_city	<i>SINGLE_OPTIONAL_STRING</i>
State	mojeid_address_bill_state	<i>SINGLE_OPTIONAL_STRING</i>
ZIP code	mojeid_address_bill_postal_code	<i>SINGLE_OPTIONAL_STRING</i>
Country	mojeid_address_bill_country	<i>SINGLE_OPTIONAL_STRING</i>
<b>Shipping address</b>		
Full address	mojeid_address_ship	<i>OPTIONAL_ADDRESS_STRING</i>
Company name	mojeid_address_ship_company_name	<i>SINGLE_OPTIONAL_STRING</i>
Street	mojeid_address_ship_street	<i>SINGLE_OPTIONAL_STRING</i>
Street 2	mojeid_address_ship_street2	<i>SINGLE_OPTIONAL_STRING</i>
Street 3	mojeid_address_ship_street3	<i>SINGLE_OPTIONAL_STRING</i>
City	mojeid_address_ship_city	<i>SINGLE_OPTIONAL_STRING</i>
State	mojeid_address_ship_state	<i>SINGLE_OPTIONAL_STRING</i>
ZIP code	mojeid_address_ship_postal_code	<i>SINGLE_OPTIONAL_STRING</i>
Country	mojeid_address_ship_country	<i>SINGLE_OPTIONAL_STRING</i>
<b>Phone</b>		
Mobile	phone_number	<i>SINGLE_OPTIONAL_STRING</i>
Flag – mobile verified ("true"/"false")	phone_number_verified	<i>SINGLE_OPTIONAL_BOOLEAN</i>
Other	mojeid_phone_mobile	<i>SINGLE_OPTIONAL_STRING</i>
Home	mojeid_phone_home	<i>SINGLE_OPTIONAL_STRING</i>
Work	mojeid_phone_office	<i>SINGLE_OPTIONAL_STRING</i>
Fax	mojeid_phone_fax	<i>SINGLE_OPTIONAL_STRING</i>
<b>Other data</b>		
Date of birth	birthdate	<i>SINGLE_OPTIONAL_STRING</i>
Gender	gender	<i>SINGLE_OPTIONAL_STRING</i>
Age	mojeid_age	<i>SINGLE_OPTIONAL_INT</i>

continues on next page

Table 1 – continued from previous page

Data	Claim identifier	Data type
ID number	mojeid_ident_card	<i>SINGLE_OPTIONAL_STRING</i>
Passport number	mojeid_ident_pass	<i>SINGLE_OPTIONAL_STRING</i>
MPSV identifier	mojeid_ident_ssn	<i>SINGLE_OPTIONAL_STRING</i>
ISIC card number <i>Only for full access</i>	mojeid_isic	<i>SINGLE_OPTIONAL_STRING</i>
Flag – older than 18 ("true"/"false")	mojeid_is_adult	<i>SINGLE_OPTIONAL_BOOLEAN</i>
Flag – student <i>Only for full access</i> ("true"/"false")	mojeid_student	<i>SINGLE_OPTIONAL_BOOLEAN</i>
Flag – validation <i>Only for full access</i> ("true"/"false")	mojeid_valid	<i>SINGLE_OPTIONAL_BOOLEAN</i>
Organization	mojeid_organization	<i>SINGLE_OPTIONAL_STRING</i>
VAT (DIČ)	mojeid_vat	<i>SINGLE_OPTIONAL_STRING</i>
VAT (IČO)	mojeid_ident_vat	<i>SINGLE_OPTIONAL_STRING</i>
Public PGP key	mojeid_public_pgp	<i>SINGLE_OPTIONAL_STRING</i>
Bank account	mojeid_bank_account	<i>SINGLE_OPTIONAL_STRING</i>
Bank account (IBAN)	mojeid_bank_account_iban	<i>SINGLE_OPTIONAL_STRING</i>
Data box	mojeid_isds	<i>SINGLE_OPTIONAL_STRING</i>
Flag - NIA <i>Only for full access</i> ("true"/"false")	mojeid_nia	<i>SINGLE_OPTIONAL_BOOLEAN</i>
<b>URL</b>		
Main	profile	<i>SINGLE_OPTIONAL_STRING</i>
Personal	website	<i>SINGLE_OPTIONAL_STRING</i>
Blog	mojeid_url_blog	<i>SINGLE_OPTIONAL_STRING</i>
Work	mojeid_url_office	<i>SINGLE_OPTIONAL_STRING</i>
RSS	mojeid_url_rss	<i>SINGLE_OPTIONAL_STRING</i>
Facebook	mojeid_url_facebook	<i>SINGLE_OPTIONAL_STRING</i>
Twitter	mojeid_url_twitter	<i>SINGLE_OPTIONAL_STRING</i>
LinkedIn	mojeid_url_linkedin	<i>SINGLE_OPTIONAL_STRING</i>
instagram	mojeid_url_instagram	<i>SINGLE_OPTIONAL_STRING</i>
pinterest	mojeid_url_pinterest	<i>SINGLE_OPTIONAL_STRING</i>
tumblr	mojeid_url_tumblr	<i>SINGLE_OPTIONAL_STRING</i>
wordpress	mojeid_url_wordpress	<i>SINGLE_OPTIONAL_STRING</i>
foursquare	mojeid_url_foursquare	<i>SINGLE_OPTIONAL_STRING</i>
youtube	mojeid_url_youtube	<i>SINGLE_OPTIONAL_STRING</i>
blogger	mojeid_url_blogger	<i>SINGLE_OPTIONAL_STRING</i>
gravatar	mojeid_url_gravatar	<i>SINGLE_OPTIONAL_STRING</i>
about_me	mojeid_url_about_me	<i>SINGLE_OPTIONAL_STRING</i>
Flickr	mojeid_url_flickr	<i>SINGLE_OPTIONAL_STRING</i>
Vimeo	mojeid_url_vimeo	<i>SINGLE_OPTIONAL_STRING</i>

continues on next page



Table 1 – continued from previous page

Data	<i>Claim identifier</i>	Data type
<b>IM</b>		
ICQ	mojeid_im_icq	<i>SINGLE_OPTIONAL_STRING</i>
Skype	mojeid_im_skype	<i>SINGLE_OPTIONAL_STRING</i>
Jabber	mojeid_im_jabber	<i>SINGLE_OPTIONAL_STRING</i>
Hangouts	mojeid_im_google_talk	<i>SINGLE_OPTIONAL_STRING</i>
Windows Live	mojeid_im_windows_live	<i>SINGLE_OPTIONAL_STRING</i>

**SINGLE\_OPTIONAL\_BOOLEAN**

Boolean or *null*

**SINGLE\_OPTIONAL\_INT**

Whole number or *null*

**SINGLE\_OPTIONAL\_STRING**

String or *null*

**OPTIONAL\_ADDRESS**

Object or *null*

Listing 1: OPTIONAL\_ADDRESS object schema

```
{
  "formatted": SINGLE_OPTIONAL_STRING,
  "street_address": SINGLE_OPTIONAL_STRING,
  "locality": SINGLE_OPTIONAL_STRING,
  "region": SINGLE_OPTIONAL_STRING,
  "postal_code": SINGLE_OPTIONAL_STRING,
  "country": SINGLE_OPTIONAL_STRING,
}
```

**OPTIONAL\_ADDRESS\_STRING**

String or *null*; string contains a serialized object *OPTIONAL\_ADDRESS*, e.g.  
"{\"formatted\": \"Sunny 5, Prague\"}"

## 9.2 Appendix 3 – List of Data to be Handed Over (SAML)

Table 2: General identifiers

Data	Identifier (URI format)	Identifier (BASIC format)
<b>Name</b>		
Whole name	urn:oid:2.5.4.3	urn:mace:dir:attribute-def:cn
First name	urn:oid:2.5.4.42	urn:mace:dir:attribute-def:givenName
Surname	urn:oid:2.5.4.4	urn:mace:dir:attribute-def:sn
Nickname	urn:oid:2.5.4.65	urn:mace:dir:attribute-def:pseudonym
<b>Email</b>		
Main	urn:oid:0.9.2342.19200300.100.1.3	urn:mace:dir:attribute-def:mail
<b>Home address</b>		
Full address	urn:oid:2.5.4.16	urn:mace:dir:attribute-def:postalAddress
Street	urn:oid:2.5.4.9	urn:mace:dir:attribute-def:street
City	urn:oid:2.5.4.7	urn:mace:dir:attribute-def:l
State	urn:oid:2.5.4.8	urn:mace:dir:attribute-def:st
Country	urn:oid:2.5.4.6	urn:mace:dir:attribute-def:c
ZIP code	urn:oid:2.5.4.17	urn:mace:dir:attribute-def:postalCode
<b>Phone</b>		
Mobile	urn:oid:2.5.4.20	urn:mace:dir:attribute-def:telephoneNumber
Fax	urn:oid:2.5.4.23	urn:mace:dir:attribute-def:facsimileTelephoneNumber
<b>Other data</b>		
Date of birth	urn:oid:1.3.6.1.4.1.2428.90.1.3	urn:mace:dir:attribute-def:norEduPersonBirthDate
Age	<a href="http://www.stork.gov.eu/1.0/age">http://www.stork.gov.eu/1.0/age</a>	
Gender	urn:oid:1.3.6.1.4.1.25178.1.2.2	
Image (base64)		urn:mace:dir:attribute-def:photo

continues on next page

Table 2 – continued from previous page

Data	Identifier (URI format)	Identifier (BASIC format)
Company name	urn:oid:2.5.4.10	urn:mace:dir:attribute-def:o
<b>URL</b>		
Main	urn:oid:1.3.6.1.4.1.27630.2.1.1.17	
Work	urn:oid:1.3.6.1.4.1.27630.2.1.1.120	

Table 3: eduID identifiers

Data	Identifier (URI format)
<b>eduID</b>	
eduPersonPrincipalName	urn:oid:1.3.6.1.4.1.5923.1.1.1.6
eduPersonScopedAffiliation	urn:oid:1.3.6.1.4.1.5923.1.1.1.9
eduPersonTargetedID	urn:oid:1.3.6.1.4.1.5923.1.1.1.10
eduPersonUniqueid	urn:oid:1.3.6.1.4.1.5923.1.1.1.13

### 9.3 Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)

Table 4: specs.nic.cz identifiers

Date	Identifier
<b>Name</b>	
Whole name	http://specs.nic.cz/attr/contact/name
First name	http://specs.nic.cz/attr/contact/name/first
Surname	http://specs.nic.cz/attr/contact/name/last
Nickname	http://specs.nic.cz/attr/contact/nickname
<b>Email</b>	
Main	http://specs.nic.cz/attr/email/main
Notify	http://specs.nic.cz/attr/email/notify
Other	http://specs.nic.cz/attr/email/next
<b>Home address</b>	
Street	http://specs.nic.cz/attr/addr/main/street
Street2	http://specs.nic.cz/attr/addr/main/street2
Street3	http://specs.nic.cz/attr/addr/main/street3
City	http://specs.nic.cz/attr/addr/main/city
State	http://specs.nic.cz/attr/addr/main/sp
Country	http://specs.nic.cz/attr/addr/main/cc
ZIP code	http://specs.nic.cz/attr/addr/main/pc
<b>Mailing address</b>	
Street	http://specs.nic.cz/attr/addr/mail/street
Street2	http://specs.nic.cz/attr/addr/mail/street2
Street3	http://specs.nic.cz/attr/addr/mail/street3
City	http://specs.nic.cz/attr/addr/mail/city
State	http://specs.nic.cz/attr/addr/mail/sp
Country	http://specs.nic.cz/attr/addr/mail/cc
ZIP code	http://specs.nic.cz/attr/addr/mail/pc
Flag – address verified <i>Only for full access</i> ("0"/"1"/"true"/"false") As of July 2022, the flag cannot be obtained for new personal accounts as they cannot be verified using PIN3.	http://specs.nic.cz/attr/addr/mail/verified
<b>Billing address</b>	
Street	http://specs.nic.cz/attr/addr/bill/street
Street2	http://specs.nic.cz/attr/addr/bill/street2
Street3	http://specs.nic.cz/attr/addr/bill/street3
City	http://specs.nic.cz/attr/addr/bill/city
State	http://specs.nic.cz/attr/addr/bill/sp
Country	http://specs.nic.cz/attr/addr/bill/cc
ZIP code	http://specs.nic.cz/attr/addr/bill/pc

continues on next page

Table 4 – continued from previous page

Date	Identifier
<b>Shipping address</b>	
Company	http://specs.nic.cz/attr/addr/ship/company_name
Street	http://specs.nic.cz/attr/addr/ship/street
Street2	http://specs.nic.cz/attr/addr/ship/street2
Street3	http://specs.nic.cz/attr/addr/ship/street3
City	http://specs.nic.cz/attr/addr/ship/city
State	http://specs.nic.cz/attr/addr/ship/sp
Country	http://specs.nic.cz/attr/addr/ship/cc
ZIP code	http://specs.nic.cz/attr/addr/ship/pc
<b>Phone</b>	
Mobile	http://specs.nic.cz/attr/phone/main
Other	http://specs.nic.cz/attr/phone/mobile
Home	http://specs.nic.cz/attr/phone/home
Work	http://specs.nic.cz/attr/phone/work
Fax	http://specs.nic.cz/attr/phone/fax
<b>Other data</b>	
Date of birth	http://specs.nic.cz/attr/contact/ident/dob
Age	http://specs.nic.cz/attr/contact/age
Gender	http://specs.nic.cz/attr/contact/gender
ID number	http://specs.nic.cz/attr/contact/ident/card
Passport number	http://specs.nic.cz/attr/contact/ident/pass
MPSV identifier	http://specs.nic.cz/attr/contact/ident/ssn
ISIC card number <i>Only for full access</i>	http://specs.nic.cz/attr/contact/isic
Flag – older than 18 ("0"/"1"/"true"/"false")	http://specs.nic.cz/attr/contact/adult
Flag – student <i>Only for full access</i> ("0"/"1"/"true"/"false")	http://specs.nic.cz/attr/contact/student
Flag – validation <i>Only for full access</i> ("0"/"1"/"true"/"false")	http://specs.nic.cz/attr/contact/valid
Account status <i>Only for full access</i>	http://specs.nic.cz/attr/contact/status
Image (base64)	http://specs.nic.cz/attr/contact/image
Company name	http://specs.nic.cz/attr/contact/org
VAT (IČO)	http://specs.nic.cz/attr/contact/ident/vat_id
VAT (DIČ)	http://specs.nic.cz/attr/contact/vat
Public PGP key	http://specs.nic.cz/attr/public_pgp
Bank account	http://specs.nic.cz/attr/bank/national
Bank account (IBAN)	http://specs.nic.cz/attr/bank/iban
Data box	http://specs.nic.cz/attr/contact/isds

continues on next page

Table 4 – continued from previous page

Date	Identifier
Flag - NIA <i>Only for full access</i> ("0"/"1"/"true"/"false")	<a href="http://specs.nic.cz/attr/contact/nia">http://specs.nic.cz/attr/contact/nia</a>
<b>Internet addresses</b>	
Main	<a href="http://specs.nic.cz/attr/url/main">http://specs.nic.cz/attr/url/main</a>
Blog	<a href="http://specs.nic.cz/attr/url/blog">http://specs.nic.cz/attr/url/blog</a>
Personal	<a href="http://specs.nic.cz/attr/url/personal">http://specs.nic.cz/attr/url/personal</a>
Work	<a href="http://specs.nic.cz/attr/url/work">http://specs.nic.cz/attr/url/work</a>
RSS	<a href="http://specs.nic.cz/attr/url/rss">http://specs.nic.cz/attr/url/rss</a>
Facebook	<a href="http://specs.nic.cz/attr/url/facebook">http://specs.nic.cz/attr/url/facebook</a>
Twitter	<a href="http://specs.nic.cz/attr/url/twitter">http://specs.nic.cz/attr/url/twitter</a>
LinkedIn	<a href="http://specs.nic.cz/attr/url/linkedin">http://specs.nic.cz/attr/url/linkedin</a>
instagram	<a href="http://specs.nic.cz/attr/url/instagram">http://specs.nic.cz/attr/url/instagram</a>
pinterest	<a href="http://specs.nic.cz/attr/url/pinterest">http://specs.nic.cz/attr/url/pinterest</a>
tumblr	<a href="http://specs.nic.cz/attr/url/tumblr">http://specs.nic.cz/attr/url/tumblr</a>
wordpress	<a href="http://specs.nic.cz/attr/url/wordpress">http://specs.nic.cz/attr/url/wordpress</a>
foursquare	<a href="http://specs.nic.cz/attr/url/foursquare">http://specs.nic.cz/attr/url/foursquare</a>
youtube	<a href="http://specs.nic.cz/attr/url/youtube">http://specs.nic.cz/attr/url/youtube</a>
blogger	<a href="http://specs.nic.cz/attr/url/blogger">http://specs.nic.cz/attr/url/blogger</a>
gravatar	<a href="http://specs.nic.cz/attr/url/gravatar">http://specs.nic.cz/attr/url/gravatar</a>
about_me	<a href="http://specs.nic.cz/attr/url/about_me">http://specs.nic.cz/attr/url/about_me</a>
Flickr	<a href="http://specs.nic.cz/attr/url/flickr">http://specs.nic.cz/attr/url/flickr</a>
Vimeo	<a href="http://specs.nic.cz/attr/url/vimeo">http://specs.nic.cz/attr/url/vimeo</a>
<b>Instant Messaging</b>	
ICQ	<a href="http://specs.nic.cz/attr/im/icq">http://specs.nic.cz/attr/im/icq</a>
Skype	<a href="http://specs.nic.cz/attr/im/skype">http://specs.nic.cz/attr/im/skype</a>
Jabber	<a href="http://specs.nic.cz/attr/im/jabber">http://specs.nic.cz/attr/im/jabber</a>
Hangouts	<a href="http://specs.nic.cz/attr/im/google_talk">http://specs.nic.cz/attr/im/google_talk</a>
Windows Live	<a href="http://specs.nic.cz/attr/im/windows_live">http://specs.nic.cz/attr/im/windows_live</a>

## 9.4 Appendix 5 – List of Data for Registration

Data	Format	Registration
<b>Name</b>		
First name	string (max 50 characters)	first_name
Surname	string (max 50 characters)	last_name
<b>Email</b>		
Main	email address (max 200 characters) <i>STD-EMAIL</i>	email_default_email
Notification	email address (max 200 characters) <i>STD-EMAIL</i>	email_notify_email
Other	email address (max 200 characters) <i>STD-EMAIL</i>	email_next_email
<b>Home address</b>		
Street	string (max 200 characters)	address_default_street1
Street2	string (max 200 characters)	address_default_street2
Street3	string (max 200 characters)	address_default_street3
City	string (max 200 characters)	address_default_city
State	string (max 200 characters)	address_default_state
ZIP code	string (max 50 characters)	address_default_postal_code
Country	kód Country podle ISO3166 <i>STD-COUNTRY</i>	address_default_country
<b>Billing address</b>		
Street	string (max 200 characters)	address_billing_street1
Street2	string (max 200 characters)	address_billing_street2
Street3	string (max 200 characters)	address_billing_street3
City	string (max 200 characters)	address_billing_city
State	string (max 200 characters)	address_billing_state
ZIP code	string (max 50 characters)	address_billing_postal_code
Country	kód Country podle ISO3166 <i>STD-COUNTRY</i>	address_billing_country

continues on next page



Table 5 – continued from previous page

Data	Format	Registration
<b>Shipping address</b>		
Company	string (max 200 characters)	address_shipping_company_name
Street	string (max 200 characters)	address_shipping_street1
Street2	string (max 200 characters)	address_shipping_street2
Street3	string (max 200 characters)	address_shipping_street3
City	string (max 200 characters)	address_shipping_city
State	string (max 200 characters)	address_shipping_state
ZIP code	string (max 50 characters)	address_shipping_postal_code
Country	kód Country podle ISO3166 <i>STD-COUNTRY</i>	address_shipping_country
<b>Mailing address</b>		
Street	string (max 200 characters)	address_mailing_street1
Street2	string (max 200 characters)	address_mailing_street2
Street3	string (max 200 characters)	address_mailing_street3
City	string (max 200 characters)	address_mailing_city
State	string (max 200 characters)	address_mailing_state
ZIP code	string (max 50 characters)	address_mailing_postal_code
Country	kód Country podle ISO3166 <i>STD-COUNTRY</i>	address_mailing_country
<b>Phone</b>		
Mobile	string that follows regular expression: ^+[0-9]{1,3}.[0-9]{1,14}\$	phone_default_number
Work	string that follows regular expression: ^+[0-9]{1,3}.[0-9]{1,14}\$	phone_office_number
Other	string that follows regular expression: ^+[0-9]{1,3}.[0-9]{1,14}\$	phone_mobile_number
Home	string that follows regular expression: ^+[0-9]{1,3}.[0-9]{1,14}\$	phone_home_number

continues on next page

Table 5 – continued from previous page

Data	Format	Registration
Phone - Fax	string that follows regular expression: ^+[-0-9]{1,3}.[-0-9]{1,14}\$	phone_fax_number
<b>Other data</b>		
Date of birth	date in the RFC3339 format (YYYY-MM-DD) <i>STD-DATE</i>	birth_date
Gender	Value "M" or "F"	gender
ID number	string (max 50 characters)	id_card_num
Passport number	string (max 50 characters)	passport_num
MPSV identifier	string (max 50 characters)	ssn_id_num
ISIC card number	string (max 50 characters)	card_isic
Company name	string (max 200 characters)	organization
VAT (İCO)	string (max 50 characters)	vat_id_num
VAT (DİC)	string (max 50 characters)	vat_reg_num
<b>Internet addresses</b>		
Main	string (max 255 characters)	urladdress_main_url
Blog	string (max 255 characters)	urladdress_blog_url
Personal	string (max 255 characters)	urladdress_personal_url
Work	string (max 255 characters)	urladdress_office_url
RSS	string (max 255 characters)	urladdress_rss_url
Facebook	string (max 255 characters)	urladdress_facebook_url
Twitter	string (max 255 characters)	urladdress_twitter_url
LinkedIn	string (max 255 characters)	urladdress_linkedin_url
instagram	string (max 255 characters)	urladdress_instagram_url
pinterest	string (max 255 characters)	urladdress_pinterest_url
tumblr	string (max 255 characters)	urladdress_tumblr_url
wordpress	string (max 255 characters)	urladdress_wordpress_url
foursquare	string (max 255 characters)	urladdress_foursquare_url

continues on next page

Table 5 – continued from previous page

Data	Format	Registration
youtube	string (max 255 characters)	urladdress_youtube_url
blogger	string (max 255 characters)	urladdress_blogger_url
gravatar	string (max 255 characters)	urladdress_gravatar_url
about_me	string (max 255 characters)	urladdress_about_me_url
<b>Instant Messaging</b>		
ICQ	string (max 255 characters)	imaccount_icq_username
Skype	string (max 255 characters)	imaccount_skype_username
Windows Live	string (max 255 characters)	imaccount_windows_live_username
Jabber	string (max 255 characters)	imaccount_jabber_username
Hangouts	string (max 255 characters)	imaccount_google_talk_username

**STD-EMAIL**

Email address in a format that follows **RFC 2822**<sup>36</sup>

**STD-COUNTRY**

Country code as per ISO 3166<sup>37</sup>

**STD-DATE**

Date in a format that follows **RFC 3339**<sup>38</sup>

<sup>36</sup> <https://datatracker.ietf.org/doc/html/rfc2822.html>

<sup>37</sup> <https://www.iso.org/iso-3166-country-codes.html>

<sup>38</sup> <https://datatracker.ietf.org/doc/html/rfc3339.html>

## 9.5 Appendix 6 – Examples and Solution of Error Messages

The following article describes the most common error messages that can occur during the implementation of MojelD. The text also provides recommendations on how to solve the issues or what to focus on.

### 9.5.1 Error Messages on Test Instance

The errors are rendered directly from the used libraries. The most important ones are described here:

- *“Error parsing document as XML”* and *“Not a XRDS document”* – Both mean there is an invalid XRDS document. This message usually describes a problem in an XRDS document with an invalid XML code (usually because it contains nonstandard unicode characters). It is possible to check the source code at <http://www.xmlvalidation.com> and find out where the error is.
- *“No XRD present in tree”* – the XRDS document has no XRD element. Check the contents of the XRDS document (see the xrd). Mind also the case of the letters inside tags!
- *“HTTP Response status from identity URL host is not 200. Got status XXX”* – a query for realm or an XRDS document returned a different HTTP status code than 200.
- Errors from cURL are in form of *“(XX, ...)”*, where XX is the number of the error from the list of libcurl errors (see <https://curl.haxx.se/libcurl/c/libcurl-errors.html>)

### 9.5.2 Problems Verifying the Return Address

When the verification of the service’s return address fails, the user is shown one of the following messages based on the phase in which the negative outcome occurred:

**a. If the connection with a service failed**

*“Nelze ověřit důvěryhodnost služby, kam se přihlašujete přes MojelD. Buďte zvláště obezřetní při předávání údajů z MojelD této službě.”*

*“We can not validate authenticity of the service where you want to login with MojelD. Use extra caution when handing over the data from MojelD.”*

This message is displayed when a query for realm or an XRDS document returns a HTTP status code 4xx or 5xx. If that is not the case, the message can describe a certificate issue when using HTTPS.

For HTTPS to work correctly, it is necessary to have a valid certificate that you can get from a certification authority (see also [Problems with Unencrypted Connection](#) (page 67)). You also need a so-called *intermediate* certificates, so that the XRDS document is searched for. The server certificate has to be set up correctly, e.g. on an Apache server, *intermediate* certificates are set up using the SSLCertificateChainFile directive or SSLCertificateFile (see [documentation for setting up SSL in Apache](#)<sup>39</sup>).

A list of certification authorities supported by MojelD can be found at [https://wiki.mozilla.org/CA/Included\\_Certificates](https://wiki.mozilla.org/CA/Included_Certificates)

---

<sup>39</sup> [https://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html#sslcertificatechainfile](https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslcertificatechainfile)

When troubleshooting issues with SSL and certificates, you can use direct tools, such as `wget` or `curl` programs, or a mechanism of a used library, that can detect issues better than common browsers.

**b. If the connection with a service was successful, but the validation of the return address failed**

*“Tento požadavek na přihlášení přes MojelD o sobě tvrdí, že přichází z jiné stránky, než tomu ve skutečnosti je. Zvažte, zda vůbec chcete pokračovat s předáváním údajů z vašeho MojelD.”*

*“This MojelD login request claims to be from other site than it really is. Consider carefully whether you want to continue with handing over the data from your MojelD.”*

Return address verification can fail due to the following reasons:

- *Realm* did not return HTTP status 200.
- There is no XRDS document on the *realm*, therefore the service cannot be verified. The XRDS document has to be placed on the *realm* in one of the three following ways:
  - The XRDS document can be placed directly in the HTTP header,
  - the XRDS document can be saved directly at the address of the *realm* (sent directly in the response),
  - the location can be described in the HTML header in a META tag.
- A redirection occurred during the downloading of the XRDS document.
- When the address `return_to` in an OpenID request does not match the address `return_to` in the XRDS document. The `return_to` address from an OpenID request can contain only several additional parameters, the so-called query string, not a subdirectory in a path.
- When the address `return_to` in an OpenID request “is not an extension” of the address of the *realm*.

The term address A “is an extension” of address B means that:

- the protocol is the same.
- the domain is the same or also contains a subdomain if the domain B starts with `*`,
- the port is the same,
- the path is the same or contains a subdirectory, and
- query string (`?key=value&key2=value2`) is the same or with additional parameters.

Table 6: Examples: the address A “is an extension” of the address B

Claim validity	Address A	Address B
yes	https://example.com/hello/	https://example.com/hello/
no	http://example.com/hello/	https://example.com/hello/
no	https://example.com:8080/hello/	https://example.com/hello/
yes	https://example.com/hello/hi/	https://example.com/hello/
no	https://example.com/hello/	https://example.com/hi/
no	https://example.com/hello/	https://example.com/hello/hi/
yes	https://example.com/hello/?key=value	https://example.com/hello/?key=value
yes	https://example.com/hello/?key=value&key2=value2	https://example.com/hello/?key=value
no	https://example.com/hello/?key=value	https://example.com/hello/?key=value&key2=value2
yes	https://subdomain.example.com/hello/?key=value	https://*.example.com/

c. **If it is not possible to manage the URL service’s area in MojelD**

*“Tento realm není dobře definovaný a nelze k němu nastavit důvěru.”*

*“This realm is not sane and thus you can not set trust for it.”*

Check that your realm (described in the identity verification request) does not contain an IP address, characters not supported in URL, or a [URI fragment](#)<sup>40</sup>. See also realm.

### 9.5.3 Problems with Unencrypted Connection

Your browser might display the following message when redirecting back to your website:

*“Informace, které jste zadali, budou odeslány přes nezašifrované spojení a mohly by jednoduše být přečteny třetí stranou. Určitě chcete pokračovat v odesílání?”*

*“The information you have entered will be sent over an unencrypted connection and could easily be read by a third party. Are you sure you want to continue sending it?”*

---

**Note:** This message comes from Firefox and it will probably look a little different in other browsers.

---

This message can appear at all *realms* without HTTPS. The data that are handed over (i.e. user’s personal information too) travel through the internet unencrypted and the browser says it leaves encrypted MojelD website towards a service that does not use encryption. We do not recommend the unencrypted protocol (HTTP), but it is possible to use it.

This issue can be solved easily by using a basic SSL certificate that can be downloaded for example here: <https://letsencrypt.org/>. The certificate secures your data transfer and at the same time, you see the level of authentication of the user.

<sup>40</sup> [https://en.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier#Generic\\_syntax](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier#Generic_syntax)

### 9.5.4 Selecting Required Logging Method

The required login method is selected by placing the identifier of the given login method into the identity verification request. The MojelD service supports not only the common login by password, but also login by a digital certificate or a one-time password (OTP).

- When logging in **using a certificate**, the following message is displayed:

*“Poskytovatel služby požaduje přihlášení certifikátem.”*

*“The service provider wants you to login with your certificate.”*

- When logging in using a **one-time password** or an **authenticator**, the following message is displayed:

*“Poskytovatel služby požaduje přihlášení jednorázovým heslem nebo MojelD Autentikátorem.”*

*“The service provider wants you to login with one time password or MojelD Autentikátor.”*

- When logging in using a **security key**, the following message is displayed:

*“Poskytovatel služby požaduje přihlášení druhým faktorem.”*

*“The service provider wants you to login with two-factor authentication.”*

Method identifiers and an example of a request with requesting a login method can be found in the `implem-oid2-zadost-overeni` section.

### 9.5.5 Problems with Library for PHP

One of the most common error messages is *“FAILED TO CREATE AUTH REQUEST: not a valid OpenID”* and *“Ověření OpenID selhalo: No OpenID information”*.

Some errors might be caused by a wrong configuration of your server. You can try to fix them in the following way:

- You need to make sure that the cURL for the given PHP version is installed, active (phpinfo should say so) and that the cURL is not disabled in `php.ini`.
- It might also be necessary to check the `/etc/php5/conf.d/curl.ini` for a line `extension=curl.so` and add it if it is not there.
- Download and install the newest version of cURL, see also <https://curl.haxx.se/download.html>.

We also recommend downloading and getting familiar with sample implementation in PHP.

### 9.5.6 Error Messages in JSON (OIDC)

Error messages contain the error code in form of an ASCII string under the `error` key. A human readable error description should be found in a JSON response under the `error_description` key.

MojelD can return the following error codes:



<b>Error Code</b>	<b>Possible Causes</b>
unauthorized_client	Wrong client_id, wrong client_secret, incorrectly used authentication.
invalid_request	Missing required parameters, one or more parameters unreadable/unparseable.

## 9.6 Appendix 7 – Correct Implementation Procedure

When implementing the MojelD service, follow these best practices:

1. Logging into the MojelD service should be initiated only by a “Login with MojelD” button, as described in the implem-oid-zadost-prihlaseni section.



2. There should be text links “Why MojelD” and “Create MojelD account” next to or under the “Login with MojelD” button.
  - a. Direct the “Why MojelD” link to a local page explaining the benefits of using MojelD on your page (local benefits) or the [information page](https://www.mojeid.cz/en/why-mojeid/)<sup>41</sup>.
  - b. The “Create MojelD account” text link can be replaced by a “Create MojelD account” button as per the example.



Direct the button to a local MojelD registration page or to a universal [MojelD registration form](https://www.mojeid.cz/registration/)<sup>42</sup>.

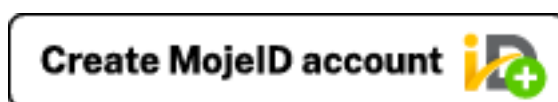
- c. If it is not possible to add links to the button as per the previous points 2.a and 2.b, we recommend to add them to an administration page of the user’s local account.
3. If possible, place a “Powered by MojelD” logo on your main page with a link to the place in your system where MojelD is used, or to the local page in your system that contains information on the MojelD service.

<sup>41</sup> <https://www.mojeid.cz/en/why-mojeid/>

<sup>42</sup> <https://mojeid.cz/registration/>



4. The data that are required to be handed over have to be in line with your system:
  - a. Only the items that are required for the registration process in your system can be marked as required.
  - b. The other items have to be marked as optional.
  - c. You must not require the disclosure of items that you do not use in your system.
5. If you require the disclosure of the user's personal data during the login using MojelD, it is recommended (in case this data differs from the data stored in the local account of your service) to let the user decide whether they want to keep the existing data in the service's local account, or whether they should be updated by the data retrieved from MojelD.
6. The implementation of the MojelD service needs to be designed in such way that the MojelD user can choose from the following two options when they first access your service using MojelD:
  - a. link MojelD with an existing local account, or
  - b. create a new local account using data retrieved from MojelD and link this newly created local account with MojelD.
7. In the user's local account administration:
  - a. We recommend to display the user's MojelD identifier upon linking with the MojelD account.
  - b. We recommend to show a link or a button "Create MojelD account" as per the point n. 2. In case the user does not have their local account linked with MojelD, and therefore probably does not have a MojelD account, we recommend to prefill the MojelD registration form with the data from the user's local account.



- c. The user needs to have an option to link MojelD with an existing local account, if it is not already linked.
  - d. The user needs to have an option to unlink the local account from MojelD.
8. Changes of the appearance of buttons and other graphical elements are possible only with an explicit consent from the CZ.NIC Association.
9. MojelD implementation must be done only using protocols OpenID Connect or SAML as per specification in the technical documentation.

**Warning:** The OpenID 2.0 protocol is no longer supported.



# Chapter 10

## Record of Changes

Version	Segment	Change description
3.1.3	<a href="#">MojelD login via PHP client</a> (page 16)	Added plugin for example login to MojelD via PHP client
3.1.2	<a href="#">Checking Data Validity</a> (page 37) <a href="#">Completing Registration</a> (page 40) <a href="#">MojelD Test Instance</a> (page 45)	Removed verification via PIN1 a PIN2
3.1.1	<a href="#">Identity verification request with a NIA-paired account</a> (page 32) <a href="#">Implementation via SAML</a> (page 33)	Fixed values of <i>acr_values</i> and <i>AuthnContextClassRef</i> including usage examples
	<a href="#">/ImplementacePodporyMojelD/OpenID Connect/KritickyMojelD</a>	Updated <i>AuthnContextClassRef</i> with NIA-paired account (CS version only)
3.1	Whole documentation	Removed all parts about OpenID 2.0 protocol from the documentation
3.0.9	<a href="#">/ImplementacePodporyMojelD/OpenID Connect/KritickyMojelD</a>	Added <i>AuthnContextClassRef</i> modules of extensions for popular platforms (CS version only)
3.0.8	<a href="#">Appendix 1 – List of Data to be Handed Over (OpenID Connect)</a> (page 50)	Added “Organization” as a handed over data Fixed VAT (IČO) and VAT (DIČ) mixup
	<a href="#">Appendix 6 – Examples and Solution of Error Messages</a> (page 64)	Changed SSL certificate link to the Let’s Encrypt service
	<a href="#">Client Registration</a> (page 21) <a href="#">MojelD LITE Library</a> (page 30)	Added note, that automatic (dynamic) registration cannot be used for Full access
3.0.7	Whole documentation	Renamed <i>mojeID</i> to <i>MojeID</i> Updated images and buttons Fixed outdated and non-working links
3.0.6	<a href="#">Appendix 1 – List of Data to be Handed Over (OpenID Connect)</a> (page 50) <a href="#">/Prilohy/UdajePredaniOID/index</a> <a href="#">Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</a> (page 57) <a href="#">Completing Registration</a> (page 40) <a href="#">MojelD Test Instance</a> (page 45)	Removed the option to verify personal account using PIN3 Added information “ <i>Only for full access</i> ” to Mailing address
3.0.5	<a href="#">Basics of MojelD</a> (page 7) <a href="#">Pairing MojelD with NIA</a> (page 14) <a href="#">Identity verification request with a NIA-paired account</a> (page 32) <a href="#">Implementation via SAML</a> (page 33)	Added information about mojelD pairing with NIA

continues on next page

Table 1 – continued from previous page

Version	Segment	Change description
<b>3.0.4</b>	<a href="#">/SeznameniSMojeid/ProcesKomunikacePresMojeid/ProcesKomunikacePresMojeid/ImplementacePodporyMojeid/Openid/index</a> <i>Favicon</i> (page 12) and all subpages <i>MojelD Test Instance</i> (page 45) <a href="#">/Prilohy/UdajePredaniOID/index</a>	Added OpenID 2.0 protocol deprecation warning on all relevant pages
<b>3.0.3</b>	<i>Appendix 1 – List of Data to be Handed Over (OpenID Connect)</i> (page 50) <a href="#">/Prilohy/UdajePredaniOID/index</a> <i>Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</i> (page 57)	Add <i>Flag - NIA</i> to the list of data to hand over
<b>3.0.2</b>	Whole documentation	Fix minor typos and untranslated text
<b>3.0.1</b>	Whole documentation	Change the term “disclose” to “hand over” in the context of the data to be handed over Fix minor typos
<b>3.0</b>	Whole documentation	Added English version of the documentation
<b>2.18</b>	<i>Appendix 1 – List of Data to be Handed Over (OpenID Connect)</i> (page 50)	Added a missing piece of data that is handed over – country (OIDC)
	<a href="#">/ImplementacePodporyMojeid/Openid/index</a> <i>Appendix 6 – Examples and Solution of Error Messages</i> (page 64)	Added a security key login method, including corresponding messages
	<i>Communication via OpenID Connect</i> (page 8) <a href="#">/SeznameniSMojeid/ProcesKomunikacePresMojeid/OpenID/index</a>	Added a login method – security key
<b>2.17</b>	<i>Client Registration</i> (page 21)	Added a part about manual registration of a client in mojelD test environment for OIDC
<b>2.16</b>	<i>Transition to a Different Protocol</i> (page 34)	Text of the Transition to a Different Protocol section changed, specific information about transition from OID2 to OIDC added
	<i>Differences Between the Protocols</i> (page 34)	Added spaces around a slash in the Implementation via OpenID Connect (OIDC) section
	<i>Implementation via OpenID Connect (OIDC)</i> (page 15)	Changed two capital letters for more consistency in the Identity Authentication Request
<b>2.15</b>	<i>Appendix 1 – List of Data to be Handed Over (OpenID Connect)</i> (page 50) <a href="#">/Prilohy/UdajePredaniOID/index</a> <i>Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</i> (page 57)	Fixed attribute For full access in the list of data to be handed over

continues on next page

Table 1 – continued from previous page

Version	Segment	Change description
2.14	<a href="#">Appendix 1 – List of Data to be Handed Over (OpenID Connect)</a> (page 50) <a href="#">/Prilohy/UdajePredaniOID/index</a>	Described data types of the data to be handed over
2.13	<a href="#">Legal Notice</a> (page 1)	Added legal notice regarding documentation
	<a href="#">Record of Changes</a>	Reworked with the newest changes on top
2.12	<a href="#">Client Registration</a> (page 21), <a href="#">Requesting Data</a> (page 29)	Fixed invalid JSON examples
	<a href="#">Appendix 1 – List of Data to be Handed Over (OpenID Connect)</a> (page 50) <a href="#">/Prilohy/UdajePredaniOID/index</a> <a href="#">Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</a> (page 57) <a href="#">Appendix 5 – List of Data for Registration</a> (page 60)	Deleted “Opencard number” and “google_plus” data to be handed over in all protocols
2.11	Everywhere	Fixed links to the new mojED website
2.10	<a href="#">Implementation Process Overview</a> (page 18)	Added implementation via OIDC process overview
	<a href="#">/ImplementacePodporyMojeid/OpenIDConnect/OverviewofModules</a>	Added Overview of Modules and modules for OIDC
	<a href="#">/ImplementacePodporyMojeid/OpenIDConnect/OverviewofLibrariesandModules</a>	Added Overview of libraries and modules for OID2
2.9	<a href="#">Favicon</a> (page 12)	Favicon’s purpose explanation and instruction for its setup
	<a href="#">Logging out of MojED</a> (page 43)	Instructions for unsubscrition
	<a href="#">Appendix 6 – Examples and Solution of Error Messages</a> (page 64)	Changed recommendation for SSL tuning
2.8	<a href="#">MojED Support Implementation</a> (page 15)	Important note on restricted usage of frameworks
2.7	<a href="#">MojED Test Instance</a> (page 45)	Updated addresses according to the new test server and explicitly listed all OIDC endpoints
	Everywhere	New technical support email address <a href="mailto:techsupport@mojeid.cz">techsupport@mojeid.cz</a>
2.6	Everywhere	Changed the order of protocols – OIDC is now the first one
	<a href="#">/ImplementacePodporyMojeid/OpenIDConnect/Interface/index</a>	Changed chapter name
2.5	<a href="#">Client Registration</a> (page 21)	Added the possibility of manual registration via OpenID Connect via the new mojED server interface
2.4	<a href="#">Interface for Creating MojED Accounts</a> (page 37)	Added support of direct registration via OpenID Connect
2.3	<a href="#">MojED Test Instance</a> (page 45)	Added information for testing communication via OIDC and SAML

continues on next page

Table 1 – continued from previous page

Version	Segment	Change description
	<a href="#">Implementation via OpenID Connect (OIDC)</a> (page 15)	Added examples of code and communication for implementation via OIDC
	<a href="#">Adjusting Communication with MojelD Server</a> (page 34)	Added recommendation for adjusting communication
	<a href="#">Appendix 6 – Examples and Solution of Error Messages</a> (page 64)	Added note on error responses in JSON for OIDC, replaced obsolete link
	<a href="#">/Prilohy/UdajePredaniOID/index</a> <a href="#">Appendix 1 – List of Data to be Handed Over (OpenID Connect)</a> (page 50) <a href="#">Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</a> (page 57)	Added a piece of data for disclosure – data box (ISDS)
2.2	<a href="#">Appendix 4 – List of Data to be Handed Over (SAML specs.nic.cz)</a> (page 57)	Added list of other identifiers for disclosure of data via SAML
2.1	<a href="#">Basics of MojelD</a> (page 7)	Moved links to protocol specifications to <a href="#">/ImplementacePodporyMojeid/Openid/index</a> and <a href="#">Implementation via OpenID Connect (OIDC)</a> (page 15)
	<a href="#">Implementation via OpenID Connect (OIDC)</a> (page 15)	Added link to configuration of OIDC on mojelD server
	<a href="#">Client Registration</a> (page 21)	Added mention of client’s metadata and extra information on manual registration
	<a href="#">MojelD LITE Library</a> (page 30)	Added a whole segment
	<a href="#">Implementation via SAML</a> (page 33)	Added a link to a certificate for verifying metadata and to a tool for decoding SAML messages
	<a href="#">Problems with Implementation</a> (page 33)	Added a whole segment
	<a href="#">Appendix 6 – Examples and Solution of Error Messages</a> (page 64)	Added a link to a tool for testing SAML settings
	Record of Changes	Added a whole segment



# Index

## A

Access Token, **6**  
Authorization Endpoint, **6**

## C

Claimed identifier, **5**  
Client ID, **6**  
Client Secret, **6**

## F

Full access, **5**

## I

ID Token, **6**  
Identifier, **5**  
Identity, **5**  
Identity name, **5**

## L

Limited access, **5**

## O

OCP, **5**  
OP, **5**  
OP endpoint, **5**  
OpenID Connect provider, **5**  
OpenID provider, **5**  
OPTIONAL\_ADDRESS, **54**  
OPTIONAL\_ADDRESS\_STRING, **54**

## R

Realm, **5**  
Refresh Token, **6**  
Registration Access Token, **6**  
Registration Endpoint, **5**  
RFC  
    RFC 2822, **63**  
    RFC 3339, **63**

## S

Service provider, **5**  
SINGLE\_OPTIONAL\_BOOLEAN, **54**  
SINGLE\_OPTIONAL\_INT, **54**  
SINGLE\_OPTIONAL\_STRING, **54**  
STD-COUNTRY, **63**  
STD-DATE, **63**  
STD-EMAIL, **63**

## T

Token Endpoint, **6**

## U

UserInfo Endpoint, **6**