Gebruikershandleiding

REST Webservice presenceRegistration

Check In and Out at Work

Versie	Datum	Aard van de wijzingen
1.0	01/03/2024	Eerste versie
1.1	03/06/2024	Herchrijving van de uitleg over OAuth

Inhoud

Aanwezigheidsregistratie bij schoonmaakactiviteiten

Conform de programmawet van 26/12/2022 moeten werknemers die specifieke schoonmaakactiviteiten uitvoeren, verplicht het begin (IN) en het einde (OUT) van hun prestaties, en ook hun rustperiodes registreren.

Een aanwezigheidsregistratie vertegenwoordigt de aankomst of het vertrek van een werknemer op zijn werkplek. Er zijn dus 2 soorten aanwezigheid:

- **IN:** bij het binnenkomen van de werknemer (maar ook bij het einde van een pauze)
- **OUT:** bij het vertrek van de werknemer (maar ook bij het begin van een pauze)

Een typische aanwezigheidsregistratie bevat de volgende informatie:

- **WIE?** Wie is het gebouw binnengekomen of eruit vertrokken? Deze informatie wordt weergegeven door het identificatienummer sociale zekerheid (INSZ) van de werknemer.
- **WAAR?** Op welke werkplek is de werknemer binnengekomen of vanwaar is die vertrokken? Deze informatie wordt weergegeven door de geografische coördinaten van de werknemer op het moment van ‘check-in’ (aankomst) of ‘check-out’ (vertrek), of door een beschrijving van het adres in tekstformaat.
- **WANNEER?** Hoe laat is de werknemer binnengekomen of vertrokken? Deze informatie wordt weergegeven door een datum en een tijdstip, waarop de werknemer zijn check-in of check-out heeft uitgevoerd.

Check **I**n **a**nd **O**ut at Work (soms afgekort als 'ClaO') is de onlinedienst waarmee de prestaties van werknemers in verschillende betrokken sectoren wordt opgevolgd. Het maakt de creatie en raadpleging van ‘presence registrations’ (aanwezigheidsregistraties) mogelijk.

Er zijn twee methoden voor het registreren van aanwezigheden:

- registratie via een smartphone, en
- beroep doen op een REST-webservice (gekoppeld aan uw eventuele bestaande badgesysteem), ook wel **PresenceRegistrations** genoemd.

Deze gebruikershandleiding beschrijft de functionele aspecten van de REST-webservice.

SOAP-service en REST-service

Het sociale zekerheidsportaal biedt twee services met de naam **PresenceRegistrations**:

- de ene service maakt gebruik van SOAP-technologie, en
- de andere van REST-technologie.

Dit zijn twee verschillende webservices, bedoeld voor specifieke sectoren:

- De **bouw**-, **vlees**- en **bewakingssectoren** moeten de **SOAP**-service gebruiken. De aanwezigheidsregistratie gebeurt eenmaal per dag, per persoon en per aangifte van werken.
- De **schoonmaaksector** moet de **REST**-service gebruiken. De aanwezigheidsregistratie gebeurt bij elke IN en OUT van de werknemer, en ook bij elke pauze voor elke werkplek.

Op dezelfde locatie kunnen er werknemers zijn die meerdere keren per dag gebruik moeten maken van de REST-service terwijl anderen slechts één keer per dag de SOAP-service zullen moeten gebruiken. Bijvoorbeeld, in het kader van een bouwplaats die ook schoonmaakdiensten omvat.Andere sectoren worden toegevoegd in de loop van 2024 en 2025. Ook zij zullen de REST-service moeten gebruiken.

Andere sectoren worden toegevoegd in de loop van 2024 en 2025. Ook zij zullen de REST-service moeten gebruiken.

Zoals hiervoor vermeld, beschrijft deze gebruiksaanwijzing enkel de REST-service. Voor de SOAP-service kunt u terecht op de pagina ‘Hoe registreert u de aanwezigheden?’ van Checkinatwork op het portaal van de sociale zekerheid.

Voorbereiding

Voordat u toegang krijgt tot de REST-service **PresenceRegistrations**, moet u een paar voorbereidende stappen uitvoeren. Als u eerder een webservice van de sociale zekerheid heeft gebruikt, is het mogelijk dat sommige van deze stappen al zijn voltooid.

Zich identificeren als werkgever

Om gebruik te maken van de onlinediensten van de sociale zekerheid moet de onderneming erkend zijn als werkgever op het portaal van de sociale zekerheid. Het portaal bundelt alle onlinediensten van de sociale zekerheid.

Om de gegevens van werkgevers en werknemers te beschermen, moet u als gebruiker van onlinediensten:

- geïdentificeerd zijn als werkgever, en
- geregistreerd zijn op het portaal.

Volg hiervoor de vier stappen die op de pagina Identificeer u als werkgever op het portaal van de sociale zekerheid.

Een certificaat aanvragen of maken

Om beroep te doen op een REST-service van de sociale zekerheid, moet u een certificaat gebruiken.

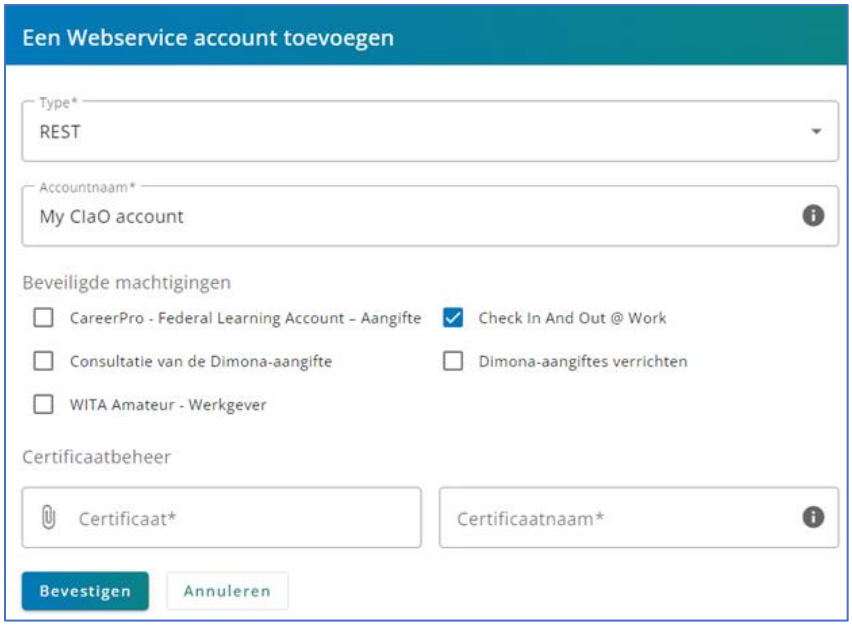
Het is niet langer verplicht om een GlobalSign-certificaat te gebruiken om toegang te krijgen tot de REST-services van de sociale zekerheid. U kunt een zelfondertekend certificaat (self-signed) gebruiken.

Webservice-account aanmaken in Chaman

Chaman of ‘Channel Management’ is de onlinedienst waarin u de technische kanalen (FTP, SFTP, SOAP en REST) van de sociale zekerheid beheert.

Een handleiding is beschikbaar op het portaal van de sociale zekerheid.

In de Chaman-onlinedienst moet je een Webservice-account van het type 'REST' aanmaken, de toestemming 'Check in And Out at Work' selecteren en daar je certificaat registreren.



- Gebruik de volgende link: [Chaman: beheer van technische kanalen]

Toegangstoken OAuth aanvragen

De REST API's van het portaal van de Sociale Zekerheid gebruiken het OAuth2-beveiligingsprotocol. Uw applicatie moet een token verkrijgen van de OAuth2-autorisatieserver van de Sociale Zekerheid.

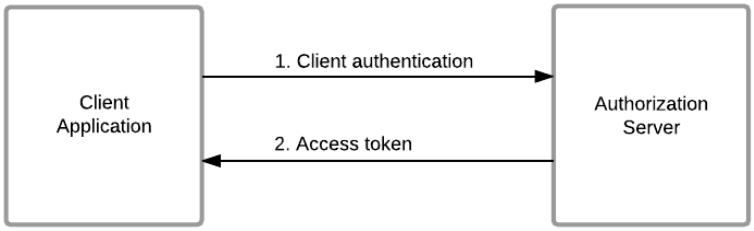
U moet hiervoor het OAuth2- client identificatieprotocol gebruiken (meer informatie op deze pagina van de IETF-website).

Om u te helpen bij uw integratie, voegen we hieronder de relevante informatie toe.

Verzoek om toegangstoken

De client authentificatie omvat de volgende stappen:

1. De client authentificeert zich bij de autorisatieserver en vraagt een toegangstoken aan bij het token-eindpunt.
2. De autorisatieserver authentificeert de client en, als deze geldig is, geeft een toegangstoken uit.



De client moet een verzoek naar het eindpunt sturen met de volgende parameters, in het formaat "**application/x-www-form-urlencoded**", en met **UTF-8** codering in de HTTP request entity-body:

Parameternaam	Verplicht	Verwachte waarde
client_assertion	Ja	Een ondertekende JWT die de client authentificeert bij de autorisatieserver zoals beschreven in RFC 7519 van de site IETF.org. De inhoud van deze JWT wordt beschreven in de sectie 'Clientauthentificatie' hieronder.
client_assertion_type	Ja	De waarde moet zijn: « **urn:ietf:params:oauth:client-assertion-type:jwt-bearer** ».
grant_type	Ja	De waarde moet zijn: « **client_credentials** ».
scope	Neen	Het betreft de scope van het toegangsverzoek. Als dit niet is ingevuld, wordt de standaard scope van uw client geselecteerd.

Het verzoek om het toegangstoken te verkrijgen moet worden ingediend bij dit eindpunt:

https://services.socialsecurity.be/REST/oauth/v5/token

Client authentificatie

Dit betreft een ondertekende JWT die de client authentificeert bij de autorisatieserver. De verwachte waarden in de payload (inhoud) van deze JWT worden beschreven in de volgende tabel.

Attribuutnaam	Verplicht	Type	Verwachte waarde
Jti	Ja	String	'jti' (JWT ID), een unieke identificatie voor de JWT
iss	Ja	String of URI	Betekent 'issuer'. De waarde moet het 'Client ID' zijn dat is gegenereerd in Chaman (*self_service_chaman_xxxxxx*).
sub	Ja	String of URI	Betekent 'subject'. De waarde moet het 'Client ID' zijn dat is gegenereerd in Chaman (*self_service_chaman_xxxxxx*).
aud	Ja	String of URI	Betekent 'audience'. De waarde moet zijn: **https://services.socialsecurity.be/REST/oauth/v5/token**
exp	Ja	Aantal (een numerieke datum in seconden)	Betekent'expiration time', m.a.w. het moment waarop de JWT niet meer moet worden geaccepteerd. De website https://unixtime.org kan u helpen om het verwachte formaat te visualiseren.
nbf	Neen	Aantal (een numerieke datum in seconden)	Betekent 'not before', m.a.w. het moment waarop de JWT moet worden geaccepteerd.
iat	Neen	Aantal (een numerieke datum in seconden)	Betekent 'issued at' m.a.w. het moment waarop de JWT is gemaakt

De handtekening moet worden aangemaakt met het certificaat (en de bijbehorende privésleutel) dat werd geüpload in Chaman.

Response

Als de authentificatie geslaagd is, ontvangt u een toegangstoken. De OAuth-server van de Sociale Zekerheid stuurt een 'Bearer'-token terug zoals beschreven in RFC 6750.

Let op: Toegangstokens van de Sociale Zekerheid hebben een geldigheidsduur van **10 minuten**.

Uw client applicatie moet dit beheren en een ander token aanvragen wanneer het huidige token bijna verloopt.

Vermijd echter het vernieuwen van het token als dat niet nodig is (wacht tot het huidige token minder dan een minuut geldig is).

Voorbeeld van Java-code

Hieronder staat een voorbeeld van Java-code met opmerkingen waarmee u een toegangstoken kunt verkrijgen van de OAuth-server van de Sociale Zekerheid.

```
package be.smals.art30bister;

import java.io.*;
import java.net.HttpURLConnection;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.security.*;
import java.util.Base64;
import java.util.Properties;
import java.util.UUID;
import java.util.logging.Level;
import java.util.logging.Logger;

public class GetAccessToken {

    private static final Logger logger =
Logger.getLogger(GetAccessToken.class.getName());

    /**
     * Main method to generate an OAuth 2.0 token using client credentials grant.
     * To run this program, provide the path to a properties file as a command line
argument.
     * The properties file should contain the following keys (and be adjusted
accordingly to your environment)
     * audience=https://services.socialsecurity.be/REST/oauth/v5/token
     * authUrl=https://services.socialsecurity.be/REST/oauth/v5/token
     * clientId=self_service_chaman_xxxxxxxxxxxxxxx  // replace with your client ID
     * scope=scope:rsz-onss:gestion:check-in-and-out-work-rest:enterprise // replace
with the scope provided in the documentation
     * password=your-key-store-password // replace with your keystore password
     * certificate=path-to-your-PKCS12-file // replace with the path to your PKCS12 file
     * keyAlias=your key alias // replace with your key alias
     *
     * @param args Command line arguments should include the path to the properties
file.
     */
    public static void main(String[] args) {
        Properties prop = new Properties();
        try (FileInputStream fis = new FileInputStream(args[0])) {
            prop.load(fis);
        } catch (IOException e) {
            logger.log(Level.SEVERE, e.getMessage(), e);
            return;
        }

        // Extract necessary properties for OAuth token request
```

```
        String audience = prop.getProperty("audience");
        String authUrl = prop.getProperty("authUrl");
        String clientId = prop.getProperty("clientId");
        String scope = prop.getProperty("scope");
        String password = prop.getProperty("password");
        String certificate = prop.getProperty("certificate");
        String keyAlias = prop.getProperty("keyAlias");

        try {
            // Generate and print the access token
            String accessTokenResponse = getAccessToken(certificate, password, keyAlias,
audience, clientId, authUrl, scope);
            logger.log(Level.INFO, accessTokenResponse);
        } catch (Exception ex) {
            logger.log(Level.SEVERE,ex.getMessage(), ex);
        }
    }

    /**
     * Retrieves the access token using the provided parameters.
     *
     * @param certificate Path to the certificate file.
     * @param password Certificate's password.
     * @param keyAlias Alias of the private key within the certificate.
     * @param audience Token endpoint audience.
     * @param clientId Client ID for OAuth.
     * @param authUrl Authorization server URL.
     * @param scope OAuth scopes requested.
     * @return A string containing the access token.
     * @throws IOException If an I/O error occurs during communication with the OAuth
server.
     * @throws GeneralSecurityException If there is an issue with the private key or
signature.
     * @throws URISyntaxException If the URI is invalid.
     */
    private static String getAccessToken(String certificate, String password, String
keyAlias, String audience, String clientId, String authUrl, String scope)
            throws IOException, GeneralSecurityException, URISyntaxException {
        KeyStore keyStore = KeyStore.getInstance("PKCS12");
        PrivateKey privateKey;

        // Load the keystore and extract the private key
        try (FileInputStream fis = new FileInputStream(certificate)) {
            keyStore.load(fis, password.toCharArray());
            privateKey = (PrivateKey) keyStore.getKey(keyAlias, password.toCharArray());
        }

        // Create a signed JWT for the OAuth authentication request
        String jwt = createJWT(privateKey, clientId, audience);

        // Send the JWT to the OAuth server and retrieve the access token
        return doPostToAuthServer(jwt, new URI(authUrl), scope);
    }

    /**
     * Creates a signed JWT for the OAuth authentication request.
```

```
     *
     * @param privateKey The private key used to sign the JWT.
     * @param clientId The OAuth client ID.
     * @param audience The audience for the token, usually the token URL.
     * @return A signed JWT string.
     * @throws NoSuchAlgorithmException If the RSA algorithm is not supported.
     * @throws InvalidKeyException If the private key is invalid.
     * @throws SignatureException If there is an issue with the signature.
     * @throws InvalidKeyException If the private key is invalid.
     */
    private static String createJWT(PrivateKey privateKey, String clientId, String
audience) throws NoSuchAlgorithmException, InvalidKeyException, SignatureException {
        long nowMillis = System.currentTimeMillis(); // Current time in milliseconds
        long expMillis = nowMillis + 3600000; // JWT validity for 1 hour
        String jti = UUID.randomUUID().toString(); // Unique identifier for each JWT

        // Create the JWT header and payload
        String header = "{\"alg\":\"RS256\",\"typ\":\"JWT\"}"; // JWT header
        String payload = String.format(

"{\"iss\":\"%s\",\"sub\":\"%s\",\"aud\":\"%s\",\"exp\":%d,\"iat\":%d,\"jti\":\"%s\"}",
                clientId, clientId, audience, expMillis / 1000, nowMillis / 1000, jti);

        // Encode the header and payload
        String encodedHeader =
Base64.getUrlEncoder().withoutPadding().encodeToString(header.getBytes(StandardCharsets.
UTF_8));
        String encodedPayload =
Base64.getUrlEncoder().withoutPadding().encodeToString(payload.getBytes(StandardCharsets
.UTF_8));

        // Concatenate the encoded header and payload
        String assertion = encodedHeader + "." + encodedPayload;

        // Sign the JWT using the RS256 algorithm
        Signature signature = Signature.getInstance("SHA256withRSA");
        signature.initSign(privateKey);
        signature.update(assertion.getBytes(StandardCharsets.UTF_8));
        byte[] signedAssertion = signature.sign();

        // Encode the signature and append it to the assertion to create the final JWT
        String encodedSignature =
Base64.getUrlEncoder().withoutPadding().encodeToString(signedAssertion);
        return assertion + "." + encodedSignature;
    }

    /**
     * Sends a POST request to the OAuth server and retrieves the access token.
     *
     * @param jwt The JWT used for client assertion.
     * @param authUri The URI of the OAuth server.
     * @param scope The requested scope(s).
     * @return The access token as a String.
     * @throws IOException If an I/O error occurs during communication with the OAuth
server.
     */
```

```
    private static String doPostToAuthServer(String jwt, URI authUri, String scope)
throws IOException {
        // Create a connection to the OAuth server
        URL url = new URL(authUri.toString());
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
        connection.setDoOutput(true);

        // Construct the request parameters
        String params =
String.format("grant_type=%s&client_assertion_type=%s&client_assertion=%s&scope=%s",
                "client_credentials", "urn:ietf:params:oauth:client-assertion-type:jwt-
bearer", jwt, scope);

        // Send the request and read the response
        try (DataOutputStream wr = new DataOutputStream(connection.getOutputStream())) {
            wr.writeBytes(params);
            wr.flush();
        }

        // Read the response from the server
        StringBuilder response = new StringBuilder();
        if (connection.getResponseCode() == HttpURLConnection.HTTP_OK) {
            try (BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()))) {
                reader.lines().forEach(response::append);
            }
        } else {
            // Read the error stream to capture any error messages from the server
            try (BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getErrorStream()))) {
                reader.lines().forEach(response::append);
            }
        }

        return response.toString();
    }


}
```

Deze code wordt verstrekt as-is. We bieden geen voorbeelden in andere programmeertalen en geven geen ondersteuning bij het schrijven van uw code.

Oproep van de REST API PresenceRegistration

Nadat u uw toegangstoken hebt ontvangen van de autorisatieserver van de Sociale Zekerheid, kunt u nu de REST-service PresenceRegistration oproepen door het ontvangen token door te geven.

Uw client applicatie moet de 'Authorization Request Header Field'-methode gebruiken zoals beschreven in sectie 2.1 van RFC 6750.

Swagger

Swagger is een bestand in YAML-formaat, een standaard voor gegevensrepresentatie. De Swagger van PresenceRegistration beschrijft de methoden en velden. U kunt dit terugvinden op het portaal van de Sociale Zekerheid, onderaan de volgende pagina:

https://www.socialsecurity.be/site_nl/employer/applics/check-in-and-out-at-work/general/how-to-register.htm

Functioneringssysteem

De RSZ stelt aan de werkgevers een webservice ter beschikking waarmee ze:

- de aanwezigheden van hun werknemers in ClaO kunnen registreren, en
- de aanwezigheden van hun werknemers en eventuele opmerkingen kunnen raadplegen.

Omgevingen

De RSZ stelt twee omgevingen ter beschikking met de volgende eindpunten:

Omgevingen	Adres
Simulatie	https://services-sim.socialsecurity.be/REST/presenceRegistration/v1
Productie	https://services.socialsecurity.be/REST/presenceRegistration/v1

Uw testen moeten worden uitgevoerd in de Simulatieomgeving.

De Productieomgeving mag alleen gegevens bevatten van daadwerkelijk uitgevoerde prestaties.

Softwareleveranciers

Dit gedeelte is bedoeld voor softwareleveranciers die hun diensten willen aanbieden aan ondernemingen waarvoor Check In & Out at Work van toepassing is.

Biedt uw dienst alleen aan om registraties naar de Sociale Zekerheid te sturen (bijvoorbeeld als u een badge-lezer beheert)?

In dat geval kunt u uw eigen certificaat gebruiken.

Wilt u geavanceerde functies aanbieden die het raadplegen van registraties en hun anomalieën omvatten?

In dat geval moet u het certificaat van uw klant gebruiken.

U biedt uw klant ondersteuning aan bij het aanmaken van een self-signed certificaat en het uploaden ervan naar Chaman.

Vervolgens kunt u uw software installeren op een machine die eigendom is van uw klant.

Asynchroniteit

Het is belangrijk op te merken dat de RSZ geen synchrone verwerking van opmerkingen rond de aanwezigheidsregistraties kan garanderen. Het verwerken van de aanwezigheden doet namelijk beroep op andere webservices, waardoor de verwerking van opmerkingen asynchroon wordt.

Daarom beschrijven de CIaO-webservices twee afzonderlijke procedures:

- het creëren van aanwezigheden, en
- het raadplegen van aanwezigheden.

Standaardverloop

Creatie

- De gebruiker moet het creatie-endpoint gebruiken om de aanwezigheden in te dienen. (**/presenceRegistrations/registerInBulk**)
- Vervolgens ontvangt u één van de volgende 2 codes:
    - **200:** in dit geval bevat het antwoord evenveel objecten als aanwezigheidsregistraties die door de gebruiker zijn aangemaakt. Voor elke registratie bevat het bijbehorende object twee elkaar uitsluitende eigenschappen :
        - **createdPresenceRegistration:** deze eigenschap beschrijft de gecreëerde aanwezigheidsregistratie. Er moet worden opgemerkt dat de eigenschap 'remarks' leeg is. Zoals eerder vermeld, is de verwerking van opmerkingen asynchroon. Daarom worden opmerkingen nog niet berekend bij het aanmaken van de aanwezigheidsregistratie in CIaO. Als deze eigenschap leeg is, geeft dit aan dat er een fout is opgetreden bij het creëren van de aanwezigheid. Deze fout wordt beschreven in de eigenschap '**notCreatedPresenceRegistration'.**
        - **notCreatedPresenceRegistration**: als deze eigenschap niet leeg is, geeft dit aan dat er een fout is opgetreden bij het creëren van de aanwezigheid. Het probleem wordt beschreven in de eigenschappen van dit object:
            - **'presenceRegistrationSubmitted'**: zoals de naam al aangeeft, bevat deze eigenschap de aanwezigheidsregistratie zoals deze door de gebruiker is ingediend. Deze aanwezigheid bevat dus één (of meerdere) fout(en), beschreven in de eigenschap '**errorList'.**
            - **'errorList'**: bevat een lijst met fouten die de redenen beschrijven waarom de registratie niet kon worden aangemaakt. Het is dus nodig om de vermelde fouten te corrigeren en **alleen de foutieve aanwezigheden opnieuw in te dienen**.
    - **500**: er heeft zich een onbeheerde fout voorgedaan, de aanwezigheidsregistraties zijn niet aangemaakt.

Raadpleging

Er zijn twee methoden om de aanwezigheidsregistraties te raadplegen:

- Op basis van een **ID**, om de informatie van een specifieke aanwezigheid te krijgen (**/presenceRegistrations/{id}**), en
- Op basis van **zoekcriteria**, om de aanwezigheden op te lijsten die overeenkomen met de genoemde criteria (**/presenceRegistration/search**).

Een gebruiker die de webservices gebruikt, zal altijd worden geassocieerd met de werkgever die het certificaat bezit. Daarom zal de raadpleging van de aanwezigheidsregistraties beperkt zijn tot de registraties voor deze werkgever, of zijn keten van onderaannemers.

Zoals eerder vermeld, zal het veld **remarks** leeg zijn bij het aanmaken van de registraties. Het is dus noodzakelijk om te wachten tot CIaO de opmerkingen heeft verwerkt vooraleer ze te proberen raadplegen.

Verwerkingstermijnen

Om onnodige belasting van de PresenceRegistration-webservice te voorkomen, is het essentieel om rekening te houden met de volgende informatie :

- Elke aanwezigheidsregistratie heeft een eigenschap **status**. Deze eigenschap geeft de status van de verwerking van opmerkingen weer. De volgende waarden zijn mogelijk:
    - **REGISTERED**: de aanwezigheid is aangemaakt, maar nog niet verwerkt, wat betekent dat de lijst met opmerkingen leeg is maar kan evolueren.
    - **VALIDATED**: de aanwezigheid is verwerkt en er is geen opmerking gegenereerd. De lijst met opmerkingen is dus leeg, en opmerkingen zullen niet meer worden berekend.
    - **CANCELLED**: de aanwezigheid is geannuleerd, en bijgevolg zullen de opmerkingen niet meer worden berekend.
    - **FAILED**: de aanwezigheid is verwerkt en er zijn opmerkingen gegenereerd. Deze lijst met opmerkingen kan in de toekomst evolueren (zie punt ‘Batch’). Let op: de aanwezigheidsregistratie is nog steeds geregistreerd. In dit geval moet u de opmerkingen controleren en beoordelen of het mogelijk is ze te corrigeren.

Status

Om controles uit te voeren op de status van de aanwezigheid, moet naar de eigenschap **code** van het object **status** worden gekeken. De **datum** geeft aan wanneer de status voor het laatst is gewijzigd:

```
...
"status": {
   "code": "registered",
   "date": "2024-02-26T15:29:22.312422275+01:00"
},
```

Wat betekent de geldigheid (validity) ‘Failed’ precies?

De geldigheid 'Failed' betekent dat de aanwezigheidsregistratie correct ontvangen is door de RSZ en succesvol is opgeslagen in onze database. Maar bij controle van de registratie zijn er opmerkingen vastgesteld die uw aandacht vereisen. De opmerkingen bevinden zich in het veld **'remarks'**.

- Een probleem met de Dimona-aangifte. In dit geval moet het probleem worden opgelost in de 'Dimona'-applicatie. Wanneer dit probleem is opgelost in Dimona, zal dit worden weergegeven in CIaO (zie het volgende punt 'Batch').
- Een probleem met de Aangifte van Werken moet worden opgelost in de 'Aangifte van werken’-applicatie. Bijvoorbeeld, de opmerking '12. Geen AVW voor het KBO'. Als u een aangevende ondernemer bent, betekent dit dat een werknemer van een onderneming die niet is aangegeven in de Aangifte van werken (AVW), voor uw aangifte heeft gewerkt. In dit geval moet u de aangifte van werken zo snel mogelijk corrigeren of de RSZ op de hoogte brengen.
- Het kan ook een probleem zijn met het gebruik van Check In and Out at Work door uw werknemers. Bijvoorbeeld, de opmerking '21. Ontbrekende registratie OUT’ betekent dat we twee opeenvolgende ‘IN’-registraties hebben ontvangen voor deze werknemer.

Batch

Een batchverwerkingssysteem wordt dagelijks uitgevoerd en herberekent opmerkingen voor aanwezigheidsregistraties met de status **REGISTERED** of **FAILED**. Elke dag zal de batch alle opmerkingen opnieuw berekenen voor de volgende datums:

Als D de datum is waarop de batch wordt uitgevoerd:

- **D - 1** (opmerkingen van de vorige dag)
- **D - 7** (opmerkingen aangemaakt 1 week geleden)
- **M - 1** (opmerkingen aangemaakt 1 maand geleden)
- **M - 3** (opmerkingen aangemaakt 3 maanden geleden)

Het is dus nutteloos om hierop elke minuut een beroep te doen voor een aanwezigheidsregistratie die bijvoorbeeld 5 dagen geleden werd aangemaakt. De opmerkingen zullen immers niet opnieuw worden verwerkt tot de volgende batch, die 2 dagen later wordt uitgevoerd: de batch ‘**D – 7’**.

Hoewel 95% van de aanwezigheidsregistraties binnen de 10 seconden wordt verwerkt (afhankelijk van het verkeer), bevelen we de volgende maxima aan qua oproepen om opmerkingen op te halen voor nieuw aangemaakte registraties:

Tijd na het aanmaken van de aanwezigheid	Frequentie van oproepen om opmerkingen op te halen
van 0 tot 1 minuut	Eén enkele oproep om de 5 seconden zolang de status van de opmerking **REGISTERED** is. Als de geldigheid (validity) staat op: • **FAILED**: de opmerkingen zullen niet meer veranderen tot de volgende dag, omdat ze zijn verwerkt • **VALIDATED**: de opmerkingen zullen niet meer veranderen omdat de aanwezigheid als geldig beschouwd wordt
Op dag D + 1	Eén enkele oproep. De opmerkingen zullen niet meer worden verwerkt tot de volgende batch (op dag D + 7), en alleen als de geldigheid van de aanwezigheid **FAILED** is.
Op dag D + 7	Eén enkele oproep. De opmerkingen worden pas weer verwerkt bij de volgende batch (op dag D + 30), en alleen als de geldigheid van de aanwezigheid **FAILED** is.
Op dag M + 1	Eén enkele oproep. De opmerkingen worden niet opnieuw verwerkt tot de volgende batch (op dag M + 1), en alleen als de geldigheid van de aanwezigheid **FAILED** is.
Op dag M + 3	Eén enkele oproep. De opmerkingen worden niet meer verwerkt, en alleen als de geldigheid van de aanwezigheid **FAILED** is.

Technische documentatie

RegisterInBulk

Beschrijving

De methode **registerInBulk** maakt het mogelijk om een of meerdere aanwezigheden te creëren (tot 200 per aanvraag). Deze aanwezigheden worden vervolgens asynchroon verwerkt door ClaO.

Aanvraag

```
HTTP POST /presenceRegistrations/registerInBulk
```

Body

De body van het verzoek moet een lijst van 'presence'-objecten bevatten, vergelijkbaar met deze :

```
{
   "items": [
       {...},
       {...}
   ]
}
```

Elk object moet overeenkomen met de onderstaande beschrijving:

Naam van de eigenschap	Type waarde	Verplicht	Beschrijving
registrationDate	string <date-time>	Ja	De datum waarop de IN of OUT heeft plaatsgevonden, in **ISO 8601-standaardformaat:** **YYYY-MM-DD*T*HH:MM:SS*Z*** • **YYYY-MM-DD** voor de datum • **HH:MM:SS** voor het uur • **Z** om de zone of tijdzone weer te geven waarin de datum wordt gegeven
ssin	string <^\d{11}$>	Ja	Het Rijksregisternummer (INSZ) van de werknemer die de IN of OUT heeft gedaan

type	string ["IN", "OUT"]	Ja	Het type aanwezigheid : • IN: begin van het werk of einde van de pauze • OUT: begin van de pauze of einde van het werk
employer	Object	Ja	De werkgever voor wie de aanwezigheid is geregistreerd. Dit object mag slechts één van deze 2 eigenschappen bevatten: • **enterpriseNumber**: **string <^[0|1]\d{9}$>**: het ondernemingsnummer van de werkgever (Belgische werkgever) • **foreignVatNumber**: **string <**maximale lengte van 255**>**: het btw-nummer van de werkgever (buitenlandse werkgever)

placeOfWork	Object	Ja	De werkplek waar de aanwezigheid is geregistreerd. Dit object mag slechts een van deze 2 eigenschappen bevatten:
			
			• **coordinates**: **object**: de coördinaten van de werkplek waar de werknemer zijn aanwezigheid registreert (in **WGS84**-formaat)
			  ○ **longitude**: **number**: de lengtegraad van de coördinaten (X)
			  ○ **latitude**: **number**: de breedtegraad van de coördinaten (Y)
			• **description** : **string <***maximale lengte van 255***>***:* vrij veld om de werkplek te beschrijven

contractualRelationshipReference	**string <^[A-HJ-NP-Z0-9]{13}$>**	Ja	De AVW-referentie van het contract tussen de klant/opdrachtgever en de aangever

Antwoord

Voor elke aanwezigheid die in de aanvraag is verzonden, is er een bijbehorend object in het antwoord:

```
{
   "items": [
        {...},
        {...}
   ]
}
```

Elk object komt overeen met de onderstaande beschrijving:

Naam van de eigenschap	Type waarde	Verplicht	Beschrijving
createdPresenceRegistration	Object	Nee	• Als er een fout is opgetreden, is deze eigenschap leeg. • Als de aanwezigheidsregistratie succesvol is opgeslagen, bevat deze eigenschap de geregistreerde aanwezigheid (met het unieke ID)
notCreatedPresenceRegistration	Object	Nee	• Als er een fout is opgetreden, heeft dit object 2 eigenschappen: ○ **presenceRegistrationSubmitted**: beschrijft de aanwezigheidsregistratie zoals deze is ingediend door de gebruiker (die dus gecorrigeerd moet worden) ○ errorList: een lijst met fouten die uitlegt waarom de aanwezigheidsregistratie niet is opgeslagen. Elke fout heeft 2 eigenschappen: ▪ **errorCode**: een technische code, uniek voor elke fout ▪ **errorDescription**: een beschrijving van de fout

Een op deze manier gecreëerde aanwezigheid heeft alle eigenschappen zoals beschreven in het endpoint '**Read by id**'.

Voorbeeld

Aanvraag

Hieronder een voorbeeld van een aanvraag met twee aanwezigheidsregistraties. De eerste is geldig, de tweede niet.

```
HTTP POST /presenceRegistrations/registerInBulk

{
  "items": [
    {
      "registrationDate": "2019-08-28T14:15:22Z",
      "ssin": "22663312345",
      "type": "in",
      "employer": {
        "enterpriseNumber": "450905686"
      },
      "placeOfWork": {
        "coordinates": {
          "longitude": 25.485606,
          "latitude": 20.673302
        }
      },
      "contractualRelationshipReference": "1Y1ZZZZZZZZZZ"
    },
    {
      "registrationDate": "2019-08-28T14:15:22Z",
      "ssin": "22663312345",
      "type": "in",
      "employer": {
        "enterpriseNumber": "4509056866666"
      },
      "placeOfWork": {
        "coordinates": {
          "longitude": 25.485606,
          "latitude": 20.673302
        }
      },
      "contractualRelationshipReference": "1Y1ZZZZZZZZZZ"
    }
  ]
}
```

Antwoord


```
[
  {
    "createdPresenceRegistration": {
      "id": 17611,
      "registrationDate": "2019-08-28T15:15:22+02:00",
      "ssin": "22663312345",
      "type": "in",
      "employer": {
        "enterpriseNumber": "450905686",
        "foreignVatNumber": null
      },
      "placeOfWork": {
        "coordinates": {
          "longitude": 25.485606,
          "latitude": 20.673302
        },
        "description": null
      },
      "contractualRelationshipReference": "1Y1ZZZZZZZZZZ",
      "activity": "cleaning",
      "channel": "ws",
      "customReference": null,
      "status": {
        "code": "registered",
        "date": "2024-02-27T07:00:33.460412229+01:00"
      },
      "validity": "pending",
      "remarks": []
    },
    "notCreatedPresenceRegistration": null
  },
  {
    "createdPresenceRegistration": null,
    "notCreatedPresenceRegistration": {
      "presenceRegistrationSubmitted": {
        "id": null,
        "registrationDate": "2019-08-28T14:15:22Z",
        "ssin": "22663312345",
        "type": "in",
        "employer": {
          "enterpriseNumber": "4509056866666",
          "foreignVatNumber": null
        },
        "placeOfWork": {
          "coordinates": {
            "longitude": 25.485606,
            "latitude": 20.673302
          },
          "description": null
        },
        "contractualRelationshipReference": "1Y1ZZZZZZZZZZ",
        "activity": null,
        "channel": null,
        "customReference": null,
        "status": null,
```

```
        "remarks": []
      },
      "errorList": [
        {
          "errorCode": "error.presence-registration.creation.enterprise-
number",
          "errorDescription": "enterprise number is not valid"
        },
        {
          "errorCode": "error.presence-registration.creation.contractual-
relationship-reference",
          "errorDescription": "contractual relationship reference is not valid"
        }
      ]
    }
  }
]
```


Toelichting

- De eerste aanwezigheidsregistratie is volledig geldig.
    - In het antwoord heeft het bijbehorende object wel degelijk een eigenschap **createdPresenceRegistration** die de gecreëerde aanwezigheid beschrijft
    - In het antwoord is de eigenschap **notCreatedPresenceRegistration** leeg, omdat er geen fout is opgetreden.
- De tweede aanwezigheidsregistratie is ongeldig.
    - Het INSZ (**SSIN**) is ongeldig.
    - Het ondernemingsnummer van de werkgever is ongeldig.
    - Daarom heeft het bijbehorende object in het antwoord een lege eigenschap **createdPresenceRegistration**.
    - Het object heeft ook een eigenschap **notCreatedPresenceRegistration**, die beschrijft :
        - **presenceRegistrationSubmitted**: de ingediende aanwezigheid met één of meerdere fouten
        - **errorList**: de lijst met fouten

Read by id

Beschrijving

Deze methode laat toe om informatie te verkrijgen over een specifieke aanwezigheid op basis van het technische ID.

Aanvraag

Waarbij **id** het ID is van de aanwezigheid waarover u informatie wilt verkrijgen:

```
HTTP GET /presenceRegistrations/{id}
```

Antwoord

Deze methode kan op verschillende manieren reageren:

Antwoordcode	Interpretatie
200	De aanwezigheid is succesvol geregistreerd.
404	• Het *id* dat in de aanvraag is vermeld, komt niet overeen met een aanwezigheid. OF • De aanwezigheid bestaat, maar is niet gekoppeld aan de werkgever (of zijn keten van onderaannemers) die is gelinkt aan de token van de aanvraag.

In geval van een code 200 bevat het antwoord dus een object zoals:

Naam van de eigenschap	Type waarde	Beschrijving
id	**number**	Unieke technische ID van de aanwezigheidsregistratie (te gebruiken in het endpoint van de raadpleging)
registrationDate	**string <date-time>**	De datum waarop de IN of OUT heeft plaatsgevonden
ssin	**string <^\d{11}$>**	Het INSZ van de werknemer die de IN of OUT heeft gedaan
worker	**Object**	Het betreft de werknemer. Dit object bevat de volgende twee eigenschappen: • givenName : **string** : de voornaam • familyName : **string** : de familienaam

type	**string ["IN", "OUT"]**	Type aanwezigheid: • IN • OUT
employer	Object	De werkgever waarvoor de aanwezigheidsregistratie is geregistreerd. Dit object mag slechts één van deze 2 eigenschappen bevatten: • **enterpriseNumber** : **string <^[0|1]\d{9}$>**: het ondernemingsnummer van de werkgever (Belgische werkgever) • **foreignVatNumber** : **string <maximale lengte 255>**: het BTW-nummer van de werkgever (buitenlandse werkgever)

placeOfWork	Object	De locatie waar de aanwezigheid is vastgelegd. Dit object mag slechts één van deze 2 eigenschappen bevatten:
		• **coordinates**: object: de coördinaten van de locatie waar de werknemer zijn aanwezigheid registreert (in het **WGS84**-formaat)
		  ○ **longitude**: **number**: de lengtegraad van de coördinaten (X)
		  ○ **latitude**: **number**: de breedtegraad van de coördinaten (Y)
		• **description**: **string <maximale lengte 255>**: een vrij veld dat de werklocatie beschrijft

contractualRelationshipReference	**string <^[A-HJ-NP-Z0-9]{13}$>**	AVW-referentie van het contract tussen de klant/opdrachtgever en de aangever
activity	**string**	Beschrijft het type activiteit dat door de werknemer is uitgevoerd
channel	**string**	Het kanaal waarmee de aanwezigheid is geregistreerd: • **MOBILE_URL** • **MOBILE_MANUAL** • **WS** • **WEB_APP**
customReference	**string**	Aanpasbare referentie om de integratie eventueel te vergemakkelijken (dit maakt het bijvoorbeeld mogelijk om een link te behouden met de technische ID van de bronapplicatie)

status	Objet	De status van de aanwezigheid. Dit object heeft 2 eigenschappen:
		• **Code**: de code van de huidige status van de aanwezigheid:
		  ○ **REGISTERED**
		  ○ **EDITED**
		  ○ **CANCELLED**
		• **date**: de datum waarop de status voor het laatst is gewijzigd

validity	Object	De geldigheid van de aanwezigheid: • **pending** • **failed** • **validated**

remarks	Lijst van objecten
	'opmerking'
		Bevat een lijst van alle opmerkingen die zijn berekend voor deze aanwezigheid. Een object 'opmerking' heeft de volgende eigenschappen:
		• **code**: de unieke code van de opmerking
		• **labels**:
		  ○ **nl**: de naam van de opmerking in het Nederlands
		  ○ **fr**: de naam van de opmerking in het Frans
		  ○ **de**: de naam van de opmerking in het Duits
		  ○ **en**: de naam van de opmerking in het Engels

Voorbeeld

Aanvraag

```
HTTP GET /presenceRegistrations/17053
```

Antwoord

```
{
      "id": 17053,
      "registrationDate": "2024-01-30T13:58:53.774+01:00",
      "ssin": "22663312345",
      "type": "in",
      "employer": {
        "enterpriseNumber": "70010100188",
        "foreignVatNumber": null
      },
```

```
      "placeOfWork": {
        "coordinates": {
          "longitude": 4.348314,
          "latitude": 50.839552
        },
        "description": null
      },
      "contractualRelationshipReference": "1Y1ZZZZZZZZZZ",
      "activity": "cleaning",
      "channel": "mobile_manual",
      "customReference": null,
      "status": {
        "code": "failed",
        "date": "2024-01-30T13:58:59.930499+01:00"
      },
      "validity": "failed",
      "remarks": [
        {
          "code": "ciao_26",
          "labels": {
            "nl": "Gps-coördinaten niet gevonden in AVW",
            "fr": "Coordonnées GPS introuvables pour la DDT",
            "de": "GPS-Koordinaten für Arbeitsmeldung nicht gefunden",
            "en": "GPS coordinates not found for DOW"
          }
        },
        {
          "code": "ciao_34",
          "labels": {
            "nl": "Registratie geweigerd door CAW, controleer de gegevens",
            "fr": "Enregistrement refusé par CAW, veuillez vérifier les
données",
            "de": "Registrierung von CAW abgelehnt, bitte überprüfen Sie die
Daten",
            "en": "Registration refused by CAW, please check data"
          }
        }
      ]
}
```

Search

Beschrijving

Deze methode laat toe om aanwezigheden op te zoeken die voldoen aan bepaalde criteria. Het resultaat is een lijst met aanwezigheidsregistraties die aan deze criteria voldoen.

Een onderneming kan opzoekingen doen voor zijn eigen werknemers of onderaannemers.

Bent u een softwareleverancier? In dat geval moet u het certificaat van uw klant gebruiken om de aanwezigheidsregistraties op te halen. Uw certificaat maakt immers de creatie van registraties mogelijk, maar niet het lezen ervan.

Aanvraag

```
HTTP POST /presenceRegistrations/search
```

Body

De **body** van het verzoek moet een object bevatten met een eigenschap **criteria**:

```
{
    "criteria": {
        ...
    }
}
```

De aanvraag moet voldoen aan de onderstaande beschrijving:

Naam van de eigenschap	Type waarde	Verplicht	Standaard waarde	Omschrijving

criteria	object	Ja		Dit zijn de zoekcriteria. Deze criteria hebben precies dezelfde eigenschappen als het object 'aanwezigheid' beschreven in het punt **Read by Id** met toevoeging van de volgende verplichte eigenschap:
				• **registrationDate**:
				  ○ **startDate:** string <date-time>: de startdatum waarop de aanwezigheidsregistraties gefilterd moeten worden
				  ○ **endDate:** string <date-time>: de einddatum waarop de aanwezigheidsregistraties gefilterd moeten worden
				
				De aanwezigheidsregistraties die resulteren uit de zoekopdracht hebben dus een registratiedatum tussen registrationDate.startDate en registrationDate.endDate

sort	object	Nee	`{`
			`"direction": "desc",`
			`"ignoreCase": false,`
			`"property": "registrationDate"`
			`}`
				Een object dat beschrijft hoe de resultaten worden gesorteerd:
				• **direction**: ["ASC", "DESC"]: om oplopend of aflopend te sorteren
				• **ignoreCase**: om aan te geven of de zoekopdracht hoofdlettergevoelig moet zijn of niet
				• **property**: de naam van de eigenschap waarop gesorteerd moet worden

Andere parameters

De aanvraag kan de volgende parameters bevatten:

Naam van de eigenschap	Type waarde	Verplicht	Standaard waarde	Omschrijving
page	number	Nee	1	De pagina waarnaar u wilt navigeren
pageSize	number	Nee	50	Het aantal items per pagina

Antwoord

Antwoordcode	Interpretatie
200	De opzoeking is succesvol verlopen, de resultaten worden getoond.
500	Er is een fout opgetreden, controleer of de zoekcriteria correct zijn geformatteerd.

In geval van een code 200, geeft het zoekresultaat een object terug volgens de volgende beschrijving:

Naam van de eigenschap	Type waarde	Omschrijving
items	Lijst van objecten	De lijst van aanwezigheden die voldoen aan de zoekcriteria. De aanwezigheden voldoen aan het formaat zoals beschreven in het punt **Read by ID.**
first	string	De link naar de eerste pagina van de zoekresultaten
last	string	De link naar de laatste pagina van de zoekresultaten
prev	string	De link naar de vorige pagina van de zoekresultaten
next	string	De link naar de volgende pagina van de zoekresultaten
page	number	Het nummer van de huidige pagina van de zoekresultaten
pageSize	number	De grootte van de huidige pagina van de zoekresultaten

sort	object	Een object dat beschrijft hoe de resultaten worden gesorteerd : • **direction**: ["ASC", "DESC"] om oplopend of aflopend te sorteren • **ignoreCase**: om aan te geven of de zoekopdracht hoofdlettergevoelig moet zijn of niet • **property**: de naam van de eigenschap waarop gesorteerd moet worden
total	number	Het totale aantal resultaten van deze zoekopdracht
totalPages	number	Het totale aantal pagina's als resultaat van deze zoekopdracht

Voorbeeld

Aanvraag

```
HTTP POST /presenceRegistrations/search

{
    "criteria": {
        "registrationDate": {
            "startDate": "2024-01-30T10:12:52+01:00",
            "endDate": "2024-02-15T10:12:54+01:00"
        },
        "type": "in"
    }
}
```

Antwoord

```
{
  "items": [
           {
           "id": 17053,
           ...
       },
       {
           "id": 17054,
           ...
       }
   ],
  "first":
"/REST/presenceRegistration/v1/presenceRegistrations/search?page=1&pageSize=50"
,
  "last":
"/REST/presenceRegistration/v1/presenceRegistrations/search?page=2&pageSize=50"
,
```

```
  "prev": null,
  "next":
"/REST/presenceRegistration/v1/presenceRegistrations/search?page=2&pageSize=50"
,
  "page": 1,
  "pageSize": 50,
  "sort": {
    "direction": "desc",
    "ignoreCase": false,
    "property": "registrationDate"
  },
  "total": 52,
  "totalPages": 2
}
```


Uitleg

Deze aanvraag geeft een lijst van alle aanwezigheidsregistraties die:

- een registratiedatum hebben tussen 30/01/2024 om 10:12 en 15/02/2024 om 12:54 (veld '**registrationDate'** in het verzoek)
- van het type 'in' (veld '**type'** in de aanvraag)

Een criterium wordt op een transparante manier toegevoegd aan de aanvraag. Deze aanwezigheidsregistraties zijn allemaal geregistreerd voor de werkgever die is gekoppeld aan het token dat wordt gebruikt in de aanvraag.