RASTIN 2.0

DATA EXCHANGE

RASTIN 2.0 SYNCHRONOUS PROTOCOL TECHNICAL SPECIFICIATION

Version of the documentation 1.1 dated 2023-05-10

Document status

Internal development

Keywords

Economic Information Bureau, Web Service, service, RASTIN, PESEL, RDO

Please send any corrections, comments and notes to pomocit@krd.pl

KRD BIG S.A., Danuty Siedzikówny 12, 51-214 Wrocław
https://en.krd.pl

Document attributes

	Attribute	Value
	A	B
1	Project	Rastin 2.0
2	Title	Data exchange
3	Subtitle	Rastin 2.0 synchronous protocol technical specification
4	Documentation version	1.1
5	Version time	2023-05-10
6	File	Rastin 2.0 - Technical specification.docx/pdf
7	Authors	Michał Łeptuch, Maciej Łukasik
8	Supervision	Justyna Grabek
9	Copyright	Copyright © Krajowy Rejestr Długów, 2023
10	Comments	

History of the document

	Attribute	Value	Date
	A	B	C
1	Documentation version	1.0	2023-04-06
2	Author	Michał Łeptuch	
3	Content proofread by	Justyna Grabek	2023-04-06
4	Form proofread by	Maciej Łukasik	2023-04-06
5	Approved by	Justyna Grabek	2023-04-06
6	Description	First version of the documentation.	
1	Documentation version	1.1	2023-05-10
2	Author	Maciej Łukasik	
3	Content proofread by	Maciej Łukasik	2023-05-10
4	Form proofread by	Maciej Łukasik	2023-05-10
5	Approved by	Maciej Łukasik	2023-05-10
6	Description	Updation in terms of inquiry (request) limits (daily and monthly).	
1	Documentation version		
2	Author		
3	Content proofread by		
4	Form proofread by		
5	Approved by		
6	Description		
1	Documentation version		
2	Author		
3	Content proofread by		
4	Form proofread by		
5	Approved by		
6	Description		

Table of Contents

Introduction

Using the services of KRD BIG S.A. (hereinafter referred to as KRD) can be done, e.g. via the bureau's website or through web services using the SOAP protocol. One of those web services is RASTIN, version 2.0.

A SOAP interface enables direct connection of the Client's application with the KRD system so that the Client directly from his / her application can synchronously (in real time, with an average response time of usually about 200-300 ms, not including network overhead, with max. timeout = 1 minute) check the data of the person being verified. This method is very convenient for Clients who can, either customize their existing applications or use the applications that have been previously pre-configured for this type of co-operation with the KRD.

This document describes the methods of the RASTIN 2.0 service and the methods of Client connection to RASTIN 2.0.

Note!

In the remainder of this document, the **RASTIN 2.0** service will be referred to as **RASTIN** (without the version number).

1. Methods of the RASTIN service

1.1. Economic information disclosure

RASTIN is a web service that inquires (requests) the PESEL register (rejestr PESEL) and the ID Card Register (Rejestr Dowodów Osobistych) which are part of the System of State Registers (System Rejestrów Państwowych, hereinafter SRP) in accordance with applicable regulations (in particular, in the *Act of 9 April 2010 on sharing economic information and exchange of economic data*, the *Act of 24.09.2010 on Population Register* and the *Act of August 6, 2010 on ID cards*) and in accordance with the technical conditions specified by the Ministry of Digitization as the owner of the registers.

With the help of the RASTIN service, the KRD Client has the opportunity to check the correctness of the submitted data before placing it in the KRD database. For this purpose, he can use the following services:

- data verification in the PESEL database
- ID card verification (information)
- ID card verification (validity)
- verification in the PESEL database if consumer is alive

2. End points

The RASTIN service to maintain compatibility with various Clients' systems exposes two end points: DefaultEndpoint and WsHttpBindingEndpoint:

- DefaultEndpoint: this end point uses the binding of the *basicHttpBinding* and is consistent with the 1.1 version of the SOAP protocol. It should be used by the Clients connecting via older systems not supporting SOAP 1.2, e.g. .NET 2.0 or older.

  The addresses of this point is:

  - production version (logging in without using a certificate): https://services.krd.pl/Rastin/v2/Verification.svc
  - demo version (logging in without using a certificate): https://demo.krd.pl/Rastin/v2/Verification.svc
  - production version (logging in using a certificate): https://services.krd.pl/Rastin/v2/cert/Verification.svc
  - demo version (logging in using a certificate): https://demo.krd.pl/Rastin/v2/cert/Verification.svc

- WsHttpBindingEndpoint: this end point is fully consistent with the SOAP 1.2 specification and WS-Addressing. It makes use of the *wsHttpBinding*. **We recommend using it**.

  The addresses of this point is:

  - production version (logging in without using a certificate): https://services.krd.pl/Rastin/v2/Verification.svc/ws
  - demo version (logging in without using a certificate): https://demo.krd.pl/Rastin/v2/Verification.svc/ws
  - production version (logging in using a certificate): https://services.krd.pl/Rastin/v2/cert/Verification.svc/ws
  - demo version (logging in using a certificate): https://demo.krd.pl/Rastin/v2/cert/Verification.svc/ws

Production WSDL: https://services.krd.pl/Rastin/v2/Verification.svc?singleWsdl
Demo WSDL: https://demo.krd.pl/Rastin/v2/Verification.svc?singleWsdl

3. Logging into the system

3.1. Logging in without using a certificate

It is possible to log into the service without using a certificate to authenticate the communication between the KRD and the Client, only through the address of the endpoint.

There are 3 options of logging in without using a certificate.

3.1.1. Login and password

To log into the system using a login and a password, set the *LoginAndPassword* value in the *Authorization* section of the *AuthorizationType* node. Additionally, enter the login in the *Login* node and the password in the *Password* node.

```
<aut:Authorization>
   <aut:AuthorizationType>LoginAndPassword</aut:AuthorizationType>
   <aut:Login>login</aut:Login>
   <aut:Password>password</aut:Password>
</aut:Authorization>
```

3.1.2. Login and encrypted password

It is *possible* to log into the service by entering a login and an encrypted password. To do this, set the *LoginAndPasswordHash* value in the *Authorization* section of the *AuthorizationType* node, enter the login in the *Login* node and the encrypted password in the *PasswordHash* node. The password hash can be calculated using the following method:

```
public static string HashPassword(string key)
{
 byte[] _key = SHA1.Create().ComputeHash(Encoding.ASCII.GetBytes(key));
 return string.Concat(_key.Select(x => x.ToString("X2")));
}
```

or:

```
public static string HashPassword(string key)
{
 byte[] _key = SHA1.Create().ComputeHash(Encoding.UTF8.GetBytes(key));
 return string.Concat(_key.Select(x => x.ToString("X2")));
}
```

```
<aut:Authorization>
   <aut:AuthorizationType>LoginAndPasswordHash</aut:AuthorizationType>
   <aut:Login>login</aut:Login>
   <aut:PasswordHash>password hash function result</aut:PasswordHash>
</aut:Authorization>
```


3.1.3. Current session ID

This method allows logging in using a previous logon ID sign string (the so-called *ticket*). It is possible if the Client has logged into the system using a different method and received a *ticket* in response. A *Ticket* remains valid for 24 hours from its creation. To log in using this method, set the *Ticket* value in the *Authorization* section of the *AuthorizationType* node and the value received during the previous logon in the *Ticket* node.

```
<aut:Authorization>
   <aut:AuthorizationType>Ticket</aut:AuthorizationType>
   <aut:Ticket>value received during the previous logon</aut:Ticket>
</aut:Authorization>
```

3.2. Logging in using a certificate

The authentication can be secured with a certificate on the part of the KRD. To use a certificate, submit an inquiry (request) to an appropriate endpoint.

The certificate is valid for one year. Please send all questions related to the certificate to pomocit@krd.pl or to your Strategic Clients Manager on the KRD side.

There are 6 options of logging in using a certificate.

3.2.1. Login and password

Enter your credentials as in logging in without a certificate.

3.2.2. Login and encrypted password

Enter your credentials as in logging in without a certificate.

3.2.3. Current session ID

Enter your credentials as in logging in without a certificate.

3.2.4. Certificate

To log in using a certificate, enter the *Certificate* value in the authorization header of the *AuthorizationType* node. The authorization will be automated for the login mapped with the certificate (usually main login) on the part of the KRD.

```
<aut:Authorization>
   <aut:AuthorizationType>Certificate</aut:AuthorizationType>
</aut:Authorization>
```


3.2.5. Certificate with credentials – logging with a different Client login

When logging in with a certificate, it is possible to confirm identity using login information mapped with the certificate (usually main login) and then to perform operations in the context of one of Client sub-logins. To that end, set the authorization type to *CertificateWithCredentials* in the *AuthorizationType* node.

Enter the Client sub-login, in the context of which operations are to be performed, in the *Login* node.

If the sub-login does not exist or has no right to perform operations, an appropriate fault (error) will be returned.

```
<aut:Authorization>
   <aut:AuthorizationType>CertificateWithCredentials</aut:AuthorizationType>
   <aut:Login>login</aut:Login>
</aut:Authorization>
```

3.2.6. Certificate with credentials – logging to a different Client's account

When logging in with a certificate, it is possible to confirm identity using login information mapped with the certificate (usually main login) and then to perform operations in the context of a different Client's account. To that end, set the authorization type to *CertificateWithCredentials* in the *AuthorizationType* node.

Enter the login credentials of a different Client's account in the *Login* node and *Password / PasswordHash* nodes as when logging in using those credentials.

If the login does not exist or has no right to perform operations, an appropriate fault (error) will be returned.

```
<aut:Authorization>
   <aut:AuthorizationType>CertificateWithCredentials</aut:AuthorizationType>
   <aut:Login>login</aut:Login>
   <aut:Password>password</aut:Password>
</aut:Authorization>

<aut:Authorization>
   <aut:AuthorizationType>CertificateWithCredentials</aut:AuthorizationType>
   <aut:Login>login</aut:Login>
   <aut:PasswordHash>password hash function result</aut:PasswordHash>
</aut:Authorization>
```


4. (*Method 1 – VerifyConsumerIdentityNumber*) PESEL number verification method

Using this method, the Client can verify the correctness and compliance of the data sent in the inquiry (request) regarding the PESEL number, name, surname and addresses of the verified person with the data appearing in the available PESEL register.

4.1. Inquiry (request) form

4.1.1. Authorization credentials

The authorization of this type of inquiry (request) is consistent with other inquiries (requests) and described in section **3**.

4.1.2. Inquiry (request) information

This section of the inquiry (request) is used to enter the identification information of the entity to verify.

VerifyConsumerIdentityNumberRequest contains 5 section (13 fields):

- *ConsumerIdentityNumber* (`string` type) – *PESEL number* of the person to verify – **required field**.

  Field validation: 11 digits
  Example: 68061200023

- *FirstName* (`string` type) – *first name* of the person to verify – **required field**.

  Field validation: from 1 to 120 characters
  Example: TERESA

- *Surname* (`string` type) – *surname* of the person to verify – **required field**.

  Field validation: from 1 to 300 characters
  Example: KOS-WIĘCEK

- *PermanentAddress* (`PermanentAddress` type) – *permanent address* of the person to verify – **optional fields**.

  Field validation: zip code only, 5 digits
  Example: 83000

  The section contains the following fields: *Street*, *Building*, *Flat*, *ZipCode* (validated), *City*

- *TemporaryAddress* (`TemporaryAddress` type) – *temporary address* of the person to verify – **optional fields**.

  Field validation: zip code only, 5 digits
  Example: 83000

  The section contains the following fields: *Street*, *Building*, *Flat*, *ZipCode* (validated), *City*

Note!

1. Data entered in the above fields are not case-sensitive.
2. If the verified person has 2 names, enter only the first name in the: *FirstName*.
3. If you enter the value -- (2 hyphens) for any element of the address section, ie: *Street*; *Building*; *Flat*; *City*, the PESEL register will not be inquired (requested). However, if your inquiries (requests) contain such values, despite the lack of actual data to be inquired (requested) for a given element, we recommend specifying hyphens for them in the inquiry (request) in an amount other than 2.
4. Definition of permanent and temporary address:
   - permanent address (*PermanentAddress*) is registered when a person registers with the intention of permanent residence,
   - temporary address (*TemporaryAddress*) is registered when a person declares to stay at that address until a certain date.

   A person can have a current permanent and temporary address at the same time. An example of such a person may be a student registered for permanent residence with parents and registered in the dormitory for the duration of the academic year.
5. Zip codes – some addresses in the PESEL register may not have zip codes (*ZipCode*), as this information (*ZipCode*) has been collected there since 2005.
6. In the *Street* field (applies to: *PermanentAddress* and *TemporaryAddress*), enter only the street name without the prefixes: **UL.**; **AL.**; **OS.** and **WS**. If there is no street in the address of the verified person, we recommend removing (deleting) the *Street* field from the inquiry (request). Removing (deleting) this field (or leaving it empty / blank) will still result in the system returning the value "**false**" in the response. The Ministry of Digitalization does not confirm with "**true**" the data that do not exist in their database. The same rule applies to e.g. addresses without the apartment / flat number, the situation in point **5.**

   Since 01-03-2015, when registering an address in the SRP, validation according to the TERYT register dictionary provided by the GUS (CSO – Central Statistical Office) is used (additionally, when entering and updating addresses, the TERYT code of the municipality is checked, specifying the province, county and municipality. TERYT codes of municipalities are published on the GUS website in the TERC file), so the street names in the PESEL register should be consistent with the names recorded in the TERYT register. It is also possible to register in the SRP an address with a street name that does not appear in the TERYT register. If the street name in the PESEL register is "Kardynała Stefana Wyszyńskiego", but in the query to the database only "Wyszyńskiego" is given, the result of the verification will be negative ("**false**") due to the inconsistency of the street name spelling. The spelling of the street name is determined by the municipality making the entry in the SRP, and in one municipality it may be, for example, "Kardynała Stefana Wyszyńskiego" and in another "Stefana Wyszyńskiego".

   ID cards issued before 01-03-2015 according to the old design can have written registered addresses, which do not have to match the addresses in the PESEL register. For an example address with an indication of the block, staircase and apartment number (**BL.5 KL.1 M.1**), **BL.5 KL.1** can be entered as the house / building number (*Building*), and **M.1** as the apartment / flat number (*Flat*). Sometimes several polling attempts (various combinations of inquiries – requests) may be necessary.
7. In the SRP it is also possible to verify foreigners. We recommend doing this with ***Method 1 – VerifyConsumerIdentityNumber*** because ***Method 2 – VerifyIDCard*** and ***Method 3 – VerifyIsIDCardCanceled*** includes only the identity cards of the Republic of Poland.

In order to verify a foreigner, it is best to specify in the request the *PESEL number* + *first name* + *surname*. A foreigner, when applying for a PESEL number does not always have to provide his permanent / temporary address, therefore answers "**false**" for address data will not always guarantee that the address given in the inquiry (request) is wrong, it may simply not exist in the PESEL register.

8. The surname entered in the PESEL register is established on the basis of a marital status certificate (marriage / birth certificate) or a document confirming the identity of the person (e.g. passport, residence card).
9. A two-part surname in the *Surname* field should usually be separated by a dash (hyphen-minus), e.g. *KOS-WIĘCEK*.
10. The SRP uses UTF-8 encoding. For the list of admissible special characters, please send a request to pomocit@krd.pl or contact your Strategic Clients Manager on the KRD side.

```
<dto:ConsumerIdentityNumberRequest>
   <dto:ConsumerIdentityNumber>68061200023</dto:ConsumerIdentityNumber>
   <dto:FirstName>TERESA</dto:FirstName>
   <dto:Surname>KOS-WIĘCEK</dto:Surname>
   <dto:PermanentAddress>
      <dto:Street>NIEPODLEGŁOŚCI</dto:Street>
      <dto:Building>2</dto:Building>
      <dto:Flat>11</dto:Flat>
      <dto:ZipCode>83000</dto:ZipCode>
      <dto:City>Pruszcz Gdański</dto:City>
   </dto:PermanentAddress>
    <dto:TemporaryAddress>
      <dto:Street>KIRKORA</dto:Street>
      <dto:Building>20</dto:Building>
      <dto:Flat>1</dto:Flat>
      <dto:ZipCode>83000</dto:ZipCode>
      <dto:City>Pruszcz Gdański</dto:City>
   </dto:TemporaryAddress>
</dto:ConsumerIdentityNumberRequest>
```

4.1.3. Response information

In response to the inquiry (request), a *VerifyConsumerIdentityNumberResponse* type element is returned, which contains 3 fields / section:

- *ConsumerIdentytyNumberIsValid* (`bool` type).
  Returns the verification result ("**true**" or "**false**") of the verified person's *PESEL number* + *first name* + *surname*,
- *PermanentAddressVerificationResult* (`AddressVerificationResult` type).
  Returns the verification result ("**true**" or "**false**") of the verified person's *permanent address*,
- *TemporaryAddressVerificationResult* (`AddressVerificationResult` type).
  Returns the verification result ("**true**" or "**false**") of the verified person's *temporary address*.

Note!

If the result of the *PESEL number* verification + *first name* + *surname* of the verified person = "**false**", in response to the inquiry (request), we receive an element of the *VerifyConsumerIdentityNumberResponse* type, containing only one field: *ConsumerIdentytyNumberIsValid*. In this situation, no further verification, i.e. verification of the *permanent* and / or *temporary address*, takes place.

4.1.4. Example inquiry (request)

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:aut="http://krd.pl/Authorization"
xmlns:dto="http://krd.pl/Rastin/Dto">
   <soap:Header>
      <aut:Authorization>
         <aut:AuthorizationType>LoginAndPassword</aut:AuthorizationType>
         <aut:Login>login</aut:Login>
         <aut:Password>password</aut:Password>
      </aut:Authorization>
   </soap:Header>
   <soap:Body>
      <dto:ConsumerIdentityNumberRequest>
         <dto:ConsumerIdentityNumber>68061200023</dto:ConsumerIdentityNumber>
         <dto:FirstName>TERESA</dto:FirstName>
         <dto:Surname>KOS-WIĘCEK</dto:Surname>
         <dto:PermanentAddress>
            <dto:Street>NIEPODLEGŁOŚCI</dto:Street>
            <dto:Building>2</dto:Building>
            <dto:Flat>11</dto:Flat>
            <dto:ZipCode>83000</dto:ZipCode>
            <dto:City>Pruszcz Gdański</dto:City>
         </dto:PermanentAddress>
          <dto:TemporaryAddress>
            <dto:Street>KIRKORA</dto:Street>
            <dto:Building>20</dto:Building>
            <dto:ZipCode>83000</dto:ZipCode>
            <dto:City>Pruszcz Gdański</dto:City>
         </dto:TemporaryAddress>
      </dto:ConsumerIdentityNumberRequest>
   </soap:Body>
</soap:Envelope>
```

4.1.5. Example response

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action
s:mustUnderstand="1">
http://krd.pl/Rastin/IVerificationService/VerifyConsumerIdentityNumberResponse
      </a:Action>
      <h:Authorization
xmlns:h="http://krd.pl/Authorization/Response"
xmlns="http://krd.pl/Authorization/Response"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Ticket>AAA1B1AA11AA111B1111AA111B1AA1111B1AA111</Ticket>
         <TicketExpirationDate>2022-02-16T12:08:59.11</TicketExpirationDate>
         <ShouldChangePassword>false</ShouldChangePassword>
         <PasswordExpirationDate>2021-05-23T00:00:00+02:00</PasswordExpirationDate>
      </h:Authorization>
   </s:Header>
```



```
   <s:Body
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <ConsumerIdentityNumberResponse
xmlns="http://krd.pl/Rastin/Dto">
         <ConsumerIdentytyNumberIsValid>true</ConsumerIdentytyNumberIsValid>
         <PermanentAddressVerificationResult>
            <StreetIsValid>true</StreetIsValid>
            <BuildingIsValid>true</BuildingIsValid>
            <FlatIsValid>true</FlatIsValid>
            <ZipCodeIsValid>true</ZipCodeIsValid>
            <CityIsValid>true</CityIsValid>
         </PermanentAddressVerificationResult>
         <TemporaryAddressVerificationResult>
            <StreetIsValid>false</StreetIsValid>
            <BuildingIsValid>false</BuildingIsValid>
            <FlatIsValid>false</FlatIsValid>
            <ZipCodeIsValid>false</ZipCodeIsValid>
            <CityIsValid>false</CityIsValid>
         </TemporaryAddressVerificationResult>
      </ConsumerIdentityNumberResponse>
   </s:Body>
</s:Envelope>
```


5. (*Method 2 – VerifyIDCard*) ID card number and series verification method

This method allows the service Client to verify the correctness and compliance of the transmitted data with the available ID Card Register.

The validity of the ID card is also verified using this method. If the verified document is invalid, while all other data provided in the inquiry (request) is correct, the system returns "**false**" for the entire inquiry (request) due to the invalidity of the ID card. In order to verify only the validity of a person's ID card, it is recommended to use a dedicated method to verify if the ID card is canceled / invalidated (***Method 3 – VerifyIsIDCardCanceled***).

5.1. Inquiry (request) form

5.1.1. Authorization credentials

The authorization of this type of inquiry (request) is consistent with other inquiries (requests) and described in section **3**.

5.1.2. Inquiry (request) information

This section of the inquiry (request) is used to enter the data indicated in the ID card which should be verified.

VerifyIDCardRequest contains 9 fields:

- *ConsumerIdentityNumber* (`string` type) – *PESEL number* – **required field**.

  Field validation: 11 digits + PESEL checksum
  Example: 69022300022

- *FirstName* (`string` type) – *first name* – **required field**.

  Field validation: from 1 to 120 characters
  Example: ZENOBIA

- *SecondName* (`string` type) – *second name* – **optional field**.

  Field validation: from 1 to 120 characters
  Example: JADWIGA

- *SurnameFirstPart* (`string` type) – *first part of the surname* – **required field**.

  Field validation: from 1 to 300 characters
  Example: PRITT

- *SurnameSecondPart* (`string` type) – *second part of the surname* – **optional field**.

  Field validation: from 1 to 300 characters
  Example: BITT

- *IDCardSeries* (`string` type) – *ID card series* – **required field**.

  Field validation: from 2 to 3 characters, A-Z range
  Example: VUH

- *IDCardNumber* (`string` type) – *ID card number* – **required field**.

  Field validation: from 6 to 7 characters, only numbers
  Example: 952097

- *ProductionDate* (`Date` type) – *ID card issued (production) date* – **required field**.

  Field validation: in the range from 1799-12-31 to 2200-01-01, date cannot be later than the date of submission of the request (but it can be according to the date of the inquiry – request)
  Example: 2014-09-24

- *ExpirationDate* (`Date` type) – *ID card expiration date* – **required field when the ID card has an expiration date**.

  Field validation: in the range from 1799-12-31 to 2200-01-01, date must be later than the date of submission of the request
  Example: 2024-09-24

  The expiry date of the ID card is **optional**. This is because ID cards, issued before 1 January 2015 to people over 65, were valid indefinitely. Since March 1, 2015, documents valid indefinitely are no longer issued. If the expiration date of the ID card is not specified, the system adds 100 years to the date of issue and inserts this date in this field to handle cases of ID cards issued indefinitely. In order to verify ID cards which, have an expiry date entered, it is necessary to enter this date.

Note!

1. Data entered in the above fields are not case-sensitive.
2. In the case of surnames consisting of two parts, we recommend passing them in one field (*SurnameFirstPart*) in the form indicated: CHRAPEK-PASEK, DE LORM etc. Sometimes several polling attempts (various combinations of inquiries – requests, e.g. using the *SurnameSecondPart* field) may be necessary. A two-part surname should usually be separated by a dash (hyphen-minus), e.g. *KOS-WIĘCEK*.
3. The SRP uses UTF-8 encoding. For the list of admissible special characters, please send a request to pomocit@krd.pl or contact your Strategic Clients Manager on the KRD side.

```
<dto:IDCardRequest>
   <dto:ConsumerIdentityNumber>69022300022</dto:ConsumerIdentityNumber>
   <dto:FirstName>ZENOBIA</dto:FirstName>
   <dto:SecondName>JADWIGA</dto:SecondName>
   <dto:SurnameFirstPart>PRITT</dto:SurnameFirstPart>
   <dto:SurnameSecondPart>BITT</dto:SurnameSecondPart>
   <dto:IDCardSeries>VUH</dto:IDCardSeries>
   <dto:IDCardNumber>952097</dto:IDCardNumber>
   <dto:ProductionDate>2014-09-24</dto:ProductionDate>
   <dto:ExpirationDate>2024-09-24</dto:ExpirationDate>
</dto:IDCardRequest>
```


5.1.3. Response information

In response to the inquiry (request), a *VerifyIDCardResponse* type element is returned, which contains 1 field:

- *IsValid* (`bool` type) – a logical value ("**true**" or "**false**") that determines if the ID card is correct.

Note!

The value "**true**" in the *IsValid* field also proves that the ID card is not canceled (invalidated), that is, it is valid.

5.1.4. Example inquiry (request)

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:aut="http://krd.pl/Authorization"
xmlns:dto="http://krd.pl/Rastin/Dto">
   <soap:Header>
      <aut:Authorization>
         <aut:AuthorizationType>Certificate</aut:AuthorizationType>
      </aut:Authorization>
   </soap:Header>
   <soap:Body>
      <dto:IDCardRequest>
         <dto:ConsumerIdentityNumber>69022300022</dto:ConsumerIdentityNumber>
         <dto:FirstName>ZENOBIA</dto:FirstName>
         <dto:SecondName>JADWIGA</dto:SecondName>
         <dto:SurnameFirstPart>PRITT</dto:SurnameFirstPart>
         <dto:SurnameSecondPart>BITT</dto:SurnameSecondPart>
         <dto:IDCardSeries>VUH</dto:IDCardSeries>
         <dto:IDCardNumber>952097</dto:IDCardNumber>
         <dto:ProductionDate>2014-09-24</dto:ProductionDate>
         <dto:ExpirationDate>2024-09-24</dto:ExpirationDate>
      </dto:IDCardRequest>
   </soap:Body>
</soap:Envelope>
```

5.1.5. Example response

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action
s:mustUnderstand="1">
http://krd.pl/Rastin/IVerificationService/VerifyIDCardResponse
      </a:Action>
      <h:Authorization
xmlns:h="http://krd.pl/Authorization/Response"
xmlns="http://krd.pl/Authorization/Response"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Ticket>AAA1B1AA11AA111B1111AA111B1AA1111B1AA111</Ticket>
         <TicketExpirationDate>2022-02-17T13:28:54.68</TicketExpirationDate>
         <ShouldChangePassword>false</ShouldChangePassword>
         <PasswordExpirationDate>2021-05-23T00:00:00+02:00</PasswordExpirationDate>
      </h:Authorization>
   </s:Header>
```



```
   <s:Body
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <IDCardResponse
xmlns="http://krd.pl/Rastin/Dto">
         <IsValid>false</IsValid>
      </IDCardResponse>
   </s:Body>
</s:Envelope>
```


6. (*Method 3* – *VerifyIsIDCardCanceled*) Canceled ID card verification method

This method allows the service Client to verify the compliance of the transmitted data with the available **list of canceled (invalidated) ID cards**. According to Article 53 of the Act of August 6, 2010 on ID cards, this is a list containing the series and numbers of canceled (invalidated) documents and the series and numbers of erroneously personalized or lost ID cards.

6.1. Inquiry (request) form

6.1.1. Authorization credentials

The authorization of this type of inquiry (request) is consistent with other inquiries (requests) and described in section **3**.

6.1.2. Inquiry (request) information

This section of the inquiry (request) is used to enter the identification information of the ID card to verify its validity.

VerifyIsIDCardCanceledRequest contains 2 fields:

- *IDCardSeries* (`string` type) – *ID card series* – **required field**.

  Field validation: from 2 to 3 characters, A-Z range
  Example: VRX

- *IDCardNumber* (`string` type) – *ID card number* – **required field**.

  Field validation: from 6 to 7 characters, only numbers
  Example: 938012

Note!

The ID card series entered in the *IDCardSeries* field is not case-sensitive.

```
<dto:IDCardCanceledRequest>
   <dto:IDCardSeries>VRX</dto:IDCardSeries>
   <dto:IDCardNumber>938012</dto:IDCardNumber>
</dto:IDCardCanceledRequest>
```

6.1.3. Response information

In response to the inquiry (request), a *VerifyIsIDCardCanceledResponse* type element is returned, which contains 1 field:

- *Canceled* (`bool` type) – a logical value ("**true**" or "**false**") specyfing if the ID card of the verified person is cancelled (invalidated).

Note!

The value "**true**" in the field *Canceled* = ID card is canceled (invalidated). The value "**false**" in the field *Canceled* = ID card is not canceled (invalidated), as long as we are 100% sure that the given series and number are consistent with the ID card.

Method 3 **–** ***VerifyIsIDCardCanceled*** returns only the information if the ID card appears in the list of canceled (invalidated) evidence.

- If in ***method 3*** the series or number of the ID card is incorrect (or both are incorrect), we get "**false**".
- If the ID card has not yet been issued and not been classified as canceled (invalid), we get "**false**".
- If the ID card has been classified as canceled (invalid), we get "**true**".

Essentially, it can be interpreted that if a given ID card series and number is not on the list of canceled (invalidated) ID cards, then the ID card is valid, but we must be sure that we are entering the correct series and number, so we cannot always do without ***method 2 – VerifyIDCard***. We recommend combining ***method 3*** with ***method 2***.

The ID card becomes a valid document when the owner picks it up from the office. According to Article 50 of the Law of August 6, 2010 on ID cards, during the period of validity, an ID card may be canceled / invalidated ("**true**" value in the *Canceled* field) in the following cases, among others:

- change of data contained in the ID card,
- change of the facial image of the ID card holder in relation to the facial image on the ID card to an extent that makes identification of the holder difficult or impossible,
- loss or damage to ID card to the extent that it makes identification of the holder difficult or impossible,
- finding the document by a third party and handing it over to the municipality or consul of the Republic of Poland,
- death of the owner – in the case of the owner's death, the ID card is automatically canceled (invalidated) after the death certificate is issued by the Registry Office,
- request for replacement of the ID card by the owner,
- loss of Polish citizenship by the owner,
- suspicion by the owner of the unauthorized use of his personal data, including the series and number of the identity card - in such case, the owner can reserve the ID card or may report this fact to the authority of any municipality in order to cancel the ID card.

6.1.4. Example inquiry (request)

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:aut="http://krd.pl/Authorization"
xmlns:dto="http://krd.pl/Rastin/Dto">
   <soap:Header>
      <aut:Authorization>
         <aut:AuthorizationType>LoginAndPassword</aut:AuthorizationType>
         <aut:Login>login</aut:Login>
         <aut:Password>password</aut:Password>
      </aut:Authorization>
   </soap:Header>
   <soap:Body>
      <dto:IDCardCanceledRequest>
         <dto:IDCardSeries>VRX</dto:IDCardSeries>
         <dto:IDCardNumber>938012</dto:IDCardNumber>
      </dto:IDCardCanceledRequest>
   </soap:Body>
</soap:Envelope>
```

6.1.5. Example response

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action
s:mustUnderstand="1">
http://krd.pl/Rastin/IVerificationService/VerifyIsIDCardCanceledResponse
      </a:Action>
      <h:Authorization
xmlns:h="http://krd.pl/Authorization/Response"
xmlns="http://krd.pl/Authorization/Response"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Ticket>AAA1B1AA11AA111B1111AA111B1AA1111B1AA111</Ticket>
         <TicketExpirationDate>2022-02-17T14:24:19.23</TicketExpirationDate>
         <ShouldChangePassword>false</ShouldChangePassword>
         <PasswordExpirationDate>2021-05-23T00:00:00+02:00</PasswordExpirationDate>
      </h:Authorization>
   </s:Header>
   <s:Body
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <IDCardCanceledResponse
xmlns="http://krd.pl/Rastin/Dto">
         <Canceled>true</Canceled>
      </IDCardCanceledResponse>
   </s:Body>
</s:Envelope>
```


7. (*Method 4* – *VerifyConsumerIsAlive*) Consumer is alive verification method

Using this method, the service Client can verify in the PESEL register if the person (based on the data sent in the request, including: PESEL number, first name, surname) is alive.

7.1. Inquiry (request) form

7.1.1. Authorization credentials

The authorization of this type of inquiry (request) is consistent with other inquiries (requests) and described in section **3**.

7.1.2. Inquiry (request) information

This section of the inquiry (request) is used to enter the identification information of the entity to verify if consumer is alive.

VerifyConsumerIsAliveRequest contains 3 fields:

- *ConsumerIdentityNumber* (`string` type) – *PESEL number* of the person to verify – **required field**.

  Field validation: 11 digits
  Example: 68061200023

- *FirstName* (`string` type) – *first name* of the person to verify – **required field**.

  Field validation: from 1 to 120 characters
  Example: TERESA

- *Surname* (`string` type) – *surname* of the person to verify – **required field**.

  Field validation: from 1 to 300 characters
  Example: KOS-WIĘCEK

Note!

1. Data entered in the above fields are not case-sensitive.
2. A two-part surname in the *Surname* field should usually be separated by a dash (hyphen-minus), e.g. *KOS-WIĘCEK*.
3. SRP uses UTF-8 encoding. For the list of admissible special characters, please send a request to pomocit@krd.pl or contact your Strategic Clients Manager on the KRD side.

```
<dto:VerifyConsumerIsAliveRequest>
   <dto:ConsumerIdentityNumber>68061200023</dto:ConsumerIdentityNumber>
   <dto:FirstName>TERESA</dto:FirstName>
   <dto:Surname>KOS-WIĘCEK</dto:Surname>
</dto:VerifyConsumerIsAliveRequest>
```


7.1.3. Response information

In response to the inquiry (request), a *VerifyConsumerIsAliveResponse* type element is returned, which contains 1 field:

- *Status* (`string` type) returns verification result, if consumer is alive:
  - **Alive** – consumer is alive,
  - **NotAlive** – consumer is not alive,
  - **IncorrectData** – given PESEL number does not exist or given first name / surname are not in line with the given PESEL number.

7.1.4. Example inquiry (request)

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:aut="http://krd.pl/Authorization"
xmlns:dto="http://krd.pl/Rastin/Dto">
   <soap:Header>
      <aut:Authorization>
         <aut:AuthorizationType>LoginAndPassword</aut:AuthorizationType>
         <aut:Login>login</aut:Login>
         <aut:Password>password</aut:Password>
      </aut:Authorization>
   </soap:Header>
   <soap:Body>
      <dto:VerifyConsumerIsAliveRequest>
         <dto:ConsumerIdentityNumber>68061200023</dto:ConsumerIdentityNumber>
         <dto:FirstName>TERESA</dto:FirstName>
         <dto:Surname>KOS-WIĘCEK</dto:Surname>
      </dto:VerifyConsumerIsAliveRequest>
   </soap:Body>
</soap:Envelope>
```

7.1.5. Example response

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action
s:mustUnderstand="1">
http://krd.pl/Rastin/IVerificationService/VerifyConsumerIsAliveResponse
      </a:Action>
      <h:Authorization
      xmlns:h="http://krd.pl/Authorization/Response"
      xmlns="http://krd.pl/Authorization/Response"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Ticket>00AF70020063538E12A4409220FC180F5A7F52000</Ticket>
         <TicketExpirationDate>2022-12-20T15:21:52+01:00</TicketExpirationDate>
         <ShouldChangePassword>false</ShouldChangePassword>
         <PasswordExpirationDate>2022-12-22T00:00:00+01:00</PasswordExpirationDate>
      </h:Authorization>
   </s:Header>
```



```
    <s:Body
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
       <ConsumerIsAliveResponse
       xmlns="http://krd.pl/Rastin/Dto">
          <Status>Alive</Status>
       </ConsumerIsAliveResponse>
    </s:Body>
 </s:Envelope>
```


8. WSDL file

8.1. Definition types

8.1.1. PermanentAddress and TemporaryAddress types

These element types are used to return the *permanent* and *temporary address* information of the person to verify in SRP. Both elements inherit from the `AddressBase` type, which consists of 5 fields:

- *Street* (`string` type) – *street*,
- *Building* (`string` type) – *house / building number*,
- *Flat* (`string` type) – *apartment / flat number*,
- *ZipCode* (`string` type) – *zip code*
  (format: ddddd, without a hyphen, e.g. 00000),
- *City* (`string` type) – *city*.

```
<dto:PermanentAddress>
   <dto:Street>NIEPODLEGŁOŚCI</dto:Street>
   <dto:Building>2</dto:Building>
   <dto:Flat>11</dto:Flat>
   <dto:ZipCode>83000</dto:ZipCode>
   <dto:City>Pruszcz Gdański</dto:City>
</dto:PermanentAddress>
 <dto:TemporaryAddress>
   <dto:Street>KIRKORA</dto:Street>
   <dto:Building>20</dto:Building>
   <dto:Flat>1</dto:Flat>
   <dto:ZipCode>83000</dto:ZipCode>
   <dto:City>Pruszcz Gdański</dto:City>
</dto:TemporaryAddress>
```


9. Faults (Errors) returned in response

If a fault (error) occurs, the response returned to the Client contains a *SOAP Fault Element* (*basicHttpBinding* / *wsHttpBinding*):

- the *faultcode* (basic) / *s:Subcode* (ws) element contains the name of the fault (error),
- the *faultstring* (basic) / *s:Reason* (ws) element is supplemented with a fault (error) message,
- the *detail* (basic) / *s:Detail* (ws) element contains the various types of fault (error) types described below.

Every method available in the RASTIN service can return one of 4 fault (error) types in the *detail* (basic) / *s:Detail* (ws) element:

- *DefaultFault*,
- *SecurityFault*,
- *ValidationFault*,
- *SchemaValidationFault*.

Example validation fault (error) response returned by RASTIN service (basic):

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Body>
      <s:Fault>
         <faultcode>s:IncorrectCredentials</faultcode>
    <faultstring xml:lang="pl-PL">Podane dane logowania są nieprawidłowe.</faultstring>
         <detail>
            <SecurityFault id="f11c0537-44da-4590-9207-03088ee3fab8" code="1"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
         </detail>
      </s:Fault>
   </s:Body>
</s:Envelope>
```

Example validation fault (error) response returned by RASTIN service (ws):

```
<s:Envelope
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://www.w3.org/2005/08/addressing">
   <s:Header>
      <a:Action
s:mustUnderstand="1">
http://krd.pl/Rastin/IVerificationService/VerifyConsumerIdentityNumber
      </a:Action>
   </s:Header>
   <s:Body>
      <s:Fault>
         <s:Code>
            <s:Value>s:Sender</s:Value>
            <s:Subcode>
               <s:Value>s:IncorrectCredentials</s:Value>
            </s:Subcode>
         </s:Code>
```



```
         <s:Reason>
            <s:Text xml:lang="pl-PL">Podane dane logowania są nieprawidłowe.</s:Text>
            <s:Text xml:lang="en-US">Specified credentials are invalid</s:Text>
         </s:Reason>
         <s:Detail>
            <SecurityFault id="bf3f81e3-1349-4b68-9c33-be8abb5897a1" code="1"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
         </s:Detail>
      </s:Fault>
   </s:Body>
</s:Envelope>
```

9.1. DefaultFault

DefaultFault is a type that is used by type *SecurityFauzlt*, *ValidationFault* and *SchemaValidationFault*, but can also occur on its own, when, for example, there is a removal (deletion) of a SOAP component (request), a renaming of some field.

This type has 2 attributes:

- *id* – fault (error) ID (`guid` type).
  It is necessary to identify the fault (error) on the side of the KRD system,
- *code* – fault (error) code (`int` type).

```
<xs:complexType name="DefaultFault">
 <xs:attribute name="id" type="q1:guid" use="required"/>
 <xs:attribute name="code" type="xs:int" use="required"/>
</xs:complexType>
```

Example:

```
            <DefaultFault id="a4ba0698-96e1-4289-8bfb-891d007b82a6" code="0"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```

9.2. SecurityFault

This type contains the *DefaultFault* type. It represents faults (errors) related to lacking rights or incorrect login credentials. It is returned when safety rules are breached.

```
<xs:complexType name="SecurityFault">
 <xs:complexContent mixed="false">
  <xs:extension base="tns:DefaultFault"/>
 </xs:complexContent>
</xs:complexType>
```

Example:

```
            <SecurityFault id="f11c0537-44da-4590-9207-03088ee3fab8" code="1"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
```


9.3. ValidationFault

This type contains the *DefaultFault* type. It represents faults (errors) related to the validation of data entered into the system (inquiry – request). This fault (error) type occurs when the entered data does not conform to the WSDL schema or does not meet the business rules (e.g. incorrect PESEL number was provided).

The *ValidationFault* type includes a *ValidationFaultDetail* element, which contains the validation fault (error) details.

```
<xs:complexType name="ValidationFault">
 <xs:complexContent mixed="false">
  <xs:extension base="tns:DefaultFault">
   <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="Details"
type="tns:ArrayOfValidationFaultDetail"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

9.3.1. ValidationFaultDetail

The *ValidationFaultDetail* type contains element:

- *Key* – key for validation fault (error) (field name),
- *Message* – validation fault (error) message.

```
<xs:complexType name="ValidationFaultDetail">
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="Key" type="xs:string"/>
  <xs:element minOccurs="0" maxOccurs="1" name="Message"
type="tns:ArrayOfValidationDetailMessageText"/>
 </xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="ValidationDetailMessageText">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute ref="xml:lang" use="optional"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

Example:

```
            <ValidationFault id="2f031f4e-4775-44fe-9867-d4b2922fcbeb" code="13"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
               <Details>
                  <Detail>
                     <Key>ConsumerIdentityNumber</Key>
                     <Message>
                        <Text xml:lang="pl-PL">Podana wartość ma nieprawidłową długość.
Musi się składać z 11 znaków.</Text>
```



```
                        <Text xml:lang="en-US">The given value has an invalid length.
Must consist of 11 characters.</Text>
                    </Message>
                </Detail>
            </Details>
        </ValidationFault>
```

9.4. SchemaValidationFault

This type contains the *DefaultFault* type. This type of error appears when the sent SOAP message does not conform to the WSDL schema (e.g. such was given: 2023-01-00 production or expiration date of ID card).

The *SchemaValidationFault* type includes a *SchemaValidationFaultDetail* element, which contains the validation fault (error) details.

```
<xs:complexType name="SchemaValidationFault">
 <xs:complexContent mixed="false">
  <xs:extension base="tns:DefaultFault">
   <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="1" name="Details"
type="tns:ArrayOfSchemaValidationFaultDetail"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

9.4.1. SchemaValidationFaultDetail

The *SchemaValidationFaultDetail* type contains 2 attributes:

- *line* – the number of the line where the error occurred,
- *code* – the number of the column where the error occurred.

```
<xs:complexType name="SchemaValidationFaultDetail">
 <xs:sequence>
  <xs:element minOccurs="0" maxOccurs="1" name="Message"
type="tns:ArrayOfValidationDetailMessageText"/>
 </xs:sequence>
 <xs:attribute name="line" type="xs:int" use="required"/>
 <xs:attribute name="column" type="xs:int" use="required"/>
</xs:complexType>
```

And element:

- *Message* – validation fault (error) message.

```
<xs:complexType name="ValidationDetailMessageText">
 <xs:simpleContent>
  <xs:extension base="xs:string">
   <xs:attribute ref="xml:lang" use="optional"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```


Example:

```
            <SchemaValidationFault id="6cbc79b1-a127-439a-ac5f-abcdd3ce8ebb" code="14"
xmlns="http://krd.pl/Faults"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
                <Details>
                   <Detail line="19" column="42">
                      <Message>
                         <Text xml:lang="pl-PL">The
'http://krd.pl/Rastin/Dto:ProductionDate' element is invalid
- The value '2023-01-00' is invalid according to its datatype
'http://www.w3.org/2001/XMLSchema:date' - The string '2023-01-00'
is not a valid Date value.</Text>
                         <Text xml:lang="en-US">The
'http://krd.pl/Rastin/Dto:ProductionDate' element is invalid
- The value '2023-01-00' is invalid according to its datatype
'http://www.w3.org/2001/XMLSchema:date' - The string '2023-01-00'
is not a valid Date value.</Text>
                      </Message>
                   </Detail>
                </Details>
             </SchemaValidationFault>
```


10. Detailed description of errors + list of errors

10.1. Code attribute

The following values and corresponding fault (error) types may appear for a code attribute:

- **0** – „**Internal service errors**”

If this type of fault (error) occurs, please check the correctness of the construction of the inquiry (request) [error triggers e.g.: mistakenly deleted SOAP component, renaming some field, specifying a different value in the authorization type (*AuthorizationType*) than the one specified in the specification / WSDL, technical break / failure on the KRD side or on the side of the Ministry of Digitization.

- **1** – „**Specified credentials are invalid**”

If this type of fault (error) occurs, please check the correctness of the login (*Login*) and password (*Password*).

- **2** – „**Insificient rights to perform specified operation**”

If this type of fault (error) occurs, please send a message to pomocit@krd.pl or to your Strategic Clients Manager on the KRD side.

- **3** – „**Monthly query limit has been reached!**”

It appears when the monthly limit of RDO / PESEL databases inquiries (requests) is exceeded (standard monthly limit = 100 000, we can increase it).

- **4** – „**The daily query limit has been reached!**”

It appears when the daily limit of RDO / PESEL databases inquiries (requests) is exceeded (standard daily limit = 10 000, we can increase it).

- **13** – „**Provided data are not valid. See details for more informations.**”

If this type of fault (error) occurs, please check the details in the element *Key* and *Message* and check the inquiry (request) for the correctness of the completed fields [error triggers e.g.: providing too short PESEL number (*ConsumerIdentityNumber*), too short ID card series (*IDCardSeries*), too short ID card number (*IDCardNumber*), ID card issue date (*ProductionDate*) later than the date of sending the inquiry (request); too short postal code (*ZipCode*); removing the required field from the inquiry (request) / leaving it empty].

- **14** – „**Provided message data are not valid. See details for more informations.**”

If this type of fault (error) occurs, please check the details in the attribute: *line*, *column* and in the element: *Message* and check the inquiry (request) for the correctness of the completed fields [error triggers e.g.: production date (*ProductionDate*) or expiration date (*ExpirationDate*) of ID card written like this: 2023-01-00; putting a letter / special character in the production date (*ProductionDate*) or expiration date (*ExpirationDate*) of the ID card, leaving these values empty; removing the field production date (*ProductionDate*) of the ID card from the inquiry (request); changing the order of the fields in the inquiry (request)].

Please send any questions related to the faults (errors) to pomocit@krd.pl or to your Strategic Clients Manager on the KRD side.

List of errors									
Method number	*faultcode / s:Subcode*	*faultstring / s:Reason*	*detail / s:Detail*	*id (ErrorId)*	*code (ErrorSubCode)*	*line*	*column*	*Key*	*Message en / en-US*
Method number	Name of the fault (error)	Fault (error) message	Type of fault (error)	Fault (error) ID (guid type)	Fault (error) code (int type)	Example of the line number where the fault (error) occurred	Example of the column number where the fault (error) occurred	Example of key for validation fault (error) (field name)	Example of validation fault (error) message
1, 2, 3, 4	s:InternalError	Internal service errors	DefaultFault	Variable	0	N/A	N/A	N/A	N/A
1, 2, 3, 4	s:IncorrectCredentials	Specified credentials are invalid	SecurityFault	Variable	1	N/A	N/A	N/A	N/A
1, 2, 3, 4	s:InsufficientRights	Insificient rights to perform specified operation	SecurityFault	Variable	2	N/A	N/A	N/A	N/A
1, 2, 3, 4	s:LimitReached	Monthly query limit has been reached!	SecurityFault	Variable	3	N/A	N/A	N/A	N/A
1, 2, 3, 4	s:LimitReached	The daily query limit has been reached!	SecurityFault	Variable	4	N/A	N/A	N/A	N/A
1, 2, 3, 4	s:DataIsNotValid	Provided data are not valid. See details for more informations.	ValidationFault	Variable	13	N/A	N/A	Details here	Details here
1, 2, 3, 4	s:MessageIsNotValid	Provided message data are not valid. See details for more informations.	SchemaValidationFault	Variable	14	34	33	N/A	The 'http://krd.pl/Rastin/Dto:ProductionDate' element is invalid - The value '2023-01-00' is invalid according to its datatype 'http://www.w3.org/2001/XMLSchema:date' – The string '2023-01-00' is not a valid Date value.

List of errors (code 13)	
Key	*Message en / en-US*
Login	Login is required.
Password	Password is required.
Authorization	Element or value are required.
ConsumerIdentityNumber	Element or value are required.
ConsumerIdentityNumber	The given value has an invalid length. Must consist of 11 characters.
ConsumerIdentityNumber	The given value has an invalid format. Only digits are permitted.
FirstName	Element or value are required.
FirstName	The given value has an invalid length. Maximum number of characters is 120.
SecondName	The given value has an invalid length. Maximum number of characters is 120.
Surname	Element or value are required.
Surname	The given value has an invalid length. Maximum number of characters is 300.
Surname (FirstPart)	Element or value are required.
Surname (FirstPart)	The given value has an invalid length. Maximum number of characters is 300.
SurnameSecondPart	The given value has an invalid length. Maximum number of characters is 300.
IDCardSeries	Element or value are required.
IDCardSeries	The given value has an invalid length. Length must be between 2 and 3 characters.
IDCardSeries	The given value has an invalid format. Must consist of 2 or 3 letters.
IDCardNumber	Element or value are required.
IDCardNumber	The given value has an invalid length. Length must be between 6 and 7 characters.
IDCardNumber	The given value has an invalid format. Only digits are permitted.
ProductionDate	Element or value are required.
ProductionDate	Date must be between 1799-12-31 and 2200-01-01.
ProductionDate	Date must be from the past.
ExpirationDate	Date must be between 1799-12-31 and 2200-01-01.
ExpirationDate	Date must be from the future.
ZipCode	Element or value are required.11
ZipCode	The given value has an invalid length. Must consist of 5 characters.
ZipCode	The given value has an invalid format. Only digits are permitted.

END OF DOCUMENT

