ORACLE®
Insurance

Oracle Insurance

Data Exchange
Integration Guide

Release 11.0.2

December 2019

ORACLE®

ORACLE®
Insurance

Contents

Preface

Welcome to the Oracle Insurance Data Exchange Integration Guide.

This guide discusses the following topics:

- Getting policy data by policy number
- Deleting policy data by policy number
- Posting update public keys

Audience

This guide is intended for users with 'OIDXOrgAdmin' role.

Related Documents

For more information, refer to the following Oracle resources:

- Oracle Insurance Data Exchange Administrators Guide
- Oracle Insurance QuickView for Life and Annuity User Guide

The documentation for OIDX is hosted online at:
https://docs.oracle.com/en/industries/insurance/insurance-data-exchange/index.html

Basic Safety Precautions

We recommend that you secure your computer when you need to step away to protect the integrity of your work.

Notational Conventions

The following text conventions are in this document:

Convention	Description
bold	**Boldface type indicates graphical user interface elements associated with an action**.
Italic	*Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.*
Monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Additional Information

For more information, please go to the Oracle Insurance web site at or contact Oracle Insurance Data Exchange support at: oidxsupport_grp@oracle.com.

Integration

Customer API (REST)

The Customer API is a RESTful service that provides the ability to retrieve, update, and/or delete information or resources owned by the customer, in compliance with the General Data Protection Regulation (GDPR) (EU) [2016/679](#).

The base URL for this service takes the following forms:

- [https://server:port/idxapi](#) accesses the latest version of the API
- [https://server:port/v.r](#) accesses a specific version of the API, where "v.r" is replaced by a version number such as "1.0".

Responses are returned as application/json, by default, but XML can be requested by setting `Accept: application/xml` in the request headers.

GET Policy Data by Policy Number

Retrieves policy data associated with the specified policy number.

Resource URL

`https://server:port/idxapi/`**`policies?polnum=`**`xxxxx`

where "xxxxx" is a policy number that identifies a policy owned by the authenticated user.

Method

GET

Authentication

HTTP Basic authentication passing the user name and password. The user must be defined in OIM with the OIDXOrgAdmin role.

Response

If the requested policy is found and successfully retrieved, the response body contains ACORD TXLife XML formatted as a string.

- If the policy is found and successfully retrieved, the HTTP response code is 200 and the response body contains a JSON object with a single property, *policyData*:

  ```
  {
      "policyData": "<TXLife><!-- ACORD CONTENT REMOVED TO SAVE
  SPACE --></TXLife>"
  }
  ```
- If the policy does not exist or does not belong to the authenticated user, the HTTP response code is 404 and the *policyData* property is "null".

  ```
  {
      "policyData": null
  }
  ```
- If the credentials used to authenticate with the REST service are not valid, the HTTP response code is 401, and the response body has the string "Unauthorized".

DELETE Policy Data by Policy Number

The DELETE Policy Data by Policy Number request works just like the GET version, except that instead of retrieving policy data, it marks the specified policy for deletion.

Resource URL

`https://server:port/idxapi/`**`policies?polnum=`**`xxxxx`

Where "xxxxx" is a policy number that identifies a policy owned by the authenticated user.

Method

DELETE

Authentication

HTTP Basic authentication passing the user name and password. The user must be defined in OIM with the OIDXOrgAdmin role.

Response

- If the operation is successful, the HTTP response code is 200 and response body contains a JSON object with the *return* property value "SUCCESS".

```
{
    "return": "SUCCESS"
}
```

- If the operation is unsuccessful due to authorization failure (HTTP response code 401), or if the specified policy is not found (HTTP response code 404), the response body contains a JSON object with the *return* property value "FAILURE".

```
{
    "return": "FAILURE"
}
```

POST Update Public Keys

Resource URL

`https://server:port/idxapi/`**`updatePublicKeys`**

Method

POST

Authentication

HTTP Basic authentication passing the user name and password. The user must be defined in OIM with the OIDXOrgAdmin role.

Request

The request body contains a JSON payload consisting of an array of *publicKeys*. Each item in the array is an object with the following properties.

- *name* – (optional) the key name associated with the public key.

- *format* – the format of the public key data.
  - o "ascii" means that the public key data is in ASCII-armored string format.

    Note: ASCII-armored JSON data must have all line breaks replaced by '\n' in the JSON string.

  - o "base64" means that the public key data has been base64 binary encoded.
- *data* – the public key file data in either ASCII-armored or base64 binary encoded format.
- *description* – (optional) description of the public key.

Following is a sample request payload that carries two public keys, one in an ASCII-armored format, and another in a base64 encoding.

> | NOTE | Do not attempt to submit the following payload on your system. The key data has been edited to save space and cannot be decoded. |

```
{
    "publicKeys": [{
        "name": "keyname1",
        "format": "ascii",
        "data": "-----BEGIN PGP PUBLIC KEY BLOCK-----\nVersion: GnuPG
v2\n\nmQENBFqHl/ABCADDpvVx9EEb6p+cYk4Dqhln+WkEnp/00jufS9rU3vFPtvd17MZT\
njnaxbbBMDbaQXQZIiZqnWxlVBo1edYP1hX977MOmJzLT/Klllyl3FwEqwkryD2Fe\nLoN4
+R+nBUaAvTH47KMGmW091/8WjQO7SvX8VHb0YvqEJg7HEz8K9nOCtrvW\njnwvfrLpTgUUiA
xUNGVLo4nB7rK6w5noz9xx7Z0LoZyTT3BCyvEy9Gk3dc+QQ8iAG\n6pIaKEU3ChnCunViAz
32L+HfH3GiCgWH6q7nF7CywCbxSp5pfPp7728x6Vvk8bhK\n8TtUOZ54EPdi\n=G4DP\n--
---END PGP PUBLIC KEY BLOCK-----",
        "description": "public key in ASCII armored format"
    },
    {
        "name": "keyname2",
        "format": "base64",
        "data":
"LS0tLS1CRUdJTiBQR1AgUFVCTElDIEtFWSBCTE9DSy0tLS0tClZlcnNpb246IEdudVBHIH
YyCgptUUVOQkZxSGwvQUJDQURECHZWeDlFRWI2cCtjWWs0RHFobG4rV2tFbnAvMDBqdWZTO
XJVM3ZGUHR2ZDE3TVpUCmpuYXhiYkJNRGJhUXhRWklpWnFuV3hsVkJvMWVkWVAxaFg5NzdN
T21KekxUL0tsbGx5bDNGdzVub3o5eHg3WjBMb1p5VFQzQkN5dkV5OUdrM2RjK1FROGlBRwo
2cElhS0VVM0NobkN1blZpQXozMkwrSGZIM0dpQ2dXSDZxN25GN0N5d0NieFNwNXBmUHA3Nz
I4eDZWdms4YmhLCjhUdFVPWjU0RVBkaQo9RzREUAotLS0tLUVORCBQR1AgUFVCTElDIEtFW
SBCTE9DSy0tLS0t",
        "description": "public key encoded as base64"
    }]
}
```

Response

- A successful response looks like this:

```
{
    "serviceStatus": "SUCCESS",
    "keyStatuses": [
```

```
        {
            "name": "keyname1",
            "status": "Success"
        },
        {
            "name": "keyname2",
            "status": "Success"
        }
    ],
    "links": [
        {
            "rel": "self",
            "href": "https://server:port/idxapi/updatePublicKeys"
        }
    ]
}
```

Here, each uploaded key has its own status property, which shows whether the associated key was successfully uploaded, or not. If all key uploads are unsuccessful, the *serviceStatus* value is "FAILURE", and if there's a mixture of failures and successes, *serviceStatus* is "PARTIAL_SUCCESS".

- If the authenticated user is not authorized to perform the operation, the HTTP response code is 401 and the response body contains a JSON object with the *serviceStatus* property value "FAILURE".

```
{
    "serviceStatus": "FAILURE"
}
```

Support

If you receive an error or need a question answered, please contact Oracle Insurance Data Exchange Support by sending an e-mail to oidxsupport_grp@oracle.com.

Address any additional inquiries to:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Index