VIRTUALIŲ PASLAUGŲ OPERATORIUS

# VPO.GateWay
# General
# V 1.2

# Table of Contents

# 1. Document summary

## 1.1. Purpose

GateWay is used for external systems or partners to interact with internal system. There are three types of interaction through GateWay: products information querying, transaction creating and transaction information querying.

GateWay interface description is divided into 3 separate documents: general part, transaction creating (together with products information querying) part and transaction information querying part.

This document describes general part of GateWay interface and communication rules and specifics how to construct XML requests and how to use HTTPS POST to send XML requests to GateWay. Using this document as a reference, you can use any programming language to write an implementation that supports communication with GateWay.

This document assumes that you are familiar with XML document design and the methods for sending data via HTTPS POST.

For other GateWay interface description documents, please see related documents.

## 1.2. Revision history

| Version | Date | Description | Author |
|---------|------|-------------|--------|
| 1.0 | 2015-10-08 | Initail version | Simonas Fergelis |
| 1.1 | 2016-02-09 | Added Terminal attribute | Simonas Fergelis |
| 1.2 | 2019-03-25 | Kind clarification | Andrius Vosylius |

## 1.3. Definitions

| Definition | Description |
|------------|-------------|
| GateWay | VPO transaction creating and quering solution |
| XML | Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable |
| HTTPS | Hypertext Transfer Protocol Secure (HTTPS) is a widely used communications protocol for secure communication over a computer network, with especially wide deployment on the Internet |
| POST | POST is one of many request methods supported by HTTP, used by the World Wide Web. The POST request method is used when the client needs to send data to the server as part of the request |
|  |  |

## 1.4. Related documents

| Document name | File name with extension |
|---------------|--------------------------|
| Vpo. GateWay. Transactions. Creating | VPO.GateWay.1.Transactions.Creating.vX.En.docx |
| Vpo. GateWay. Transactions. Quering | VPO.GateWay.2.Transactions.Quering.vX.En.docx |
|  |  |

# 2. Getting started

GateWay is used for external systems or partners to interact with internal system. GateWay accepts XML messages that are posted over HTTPS protocol.

GateWay assumes that the client process that is requesting an action waits for a result from the server process in response to the requested action (synchronous operation). The protocol does not support session tracking.

Types of interaction with GateWay are shown below:
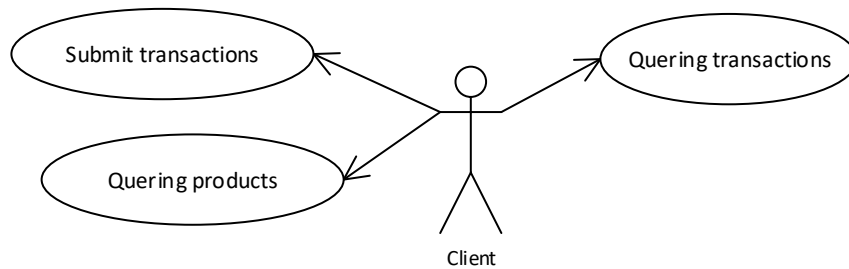


**Figure 1**: GateWay. Types of interaction

This document focuses on general communication with GateWay aspects. Same principles are used for all types of interaction. For detailed description of interaction commands refer to related.

# 3. Basic XML structure

## 3.1. Request

All requests that are sent to GateWay have main XML structure. Remember that data are case-sensitive. Structure is defined below:

```xml
<Request>
    <Data RequestId="1" Source="Source">
        <!-- Command XML data-->
    </Data>
    <Signature>
        atAozKiFPZXSRRWTewcTnV8NvDkxyXylVPmoLIG8HgBBg7eviIc15dkRePKP6OGDwCRtbvxwn8qLN76
        l8YivftMLXW9yw4cINDzFVNkWvHWOJ+2obKd1+DyVEUdOcNYes8p1n/Ns5fmPnPLPVo6Td2uqRM8gh
        KbbxiSi+RROrH1e6ktINVTLJsrOO3WiFjdhEckMdnKT+xbWj0zz38Nm08KTtXRdJLHx939JjJ8YjWGKlrW
        +eQLkvX1LGF6hXKXQqrtCHPqzURYaPsFs/oTzit5De52NzryW8Ts51B3QJeX3WPvGiL3kXx0cAkYS+T4
        LxPrFM992PXHJvG3LjuFnOw==
    </Signature>
</Request>
```

Commands XML data are described in section "Commands".

There are two nodes in root tag: Data and Signature. Data node stores information about the request. Inner Data node xml depends on action what needs to be executed. Signature value is signed **Data** node in BASE64 encoding. It is very important to consider possible differences and maintain the right format and encoding of XML for signature calculation (more details in "4.2 "Data" XML format in request message and encoding for signature").

Data attributes description:

| Name | Type | Mandatory | Description |
|---|---|---|---|
| RequestId | String(64) | Yes | External system request unique identifier. GUID might be used |
| Source | String(200) | Yes | External system unique identifier |

Unique identifier ("Source") is issued to every external system. External system provides this identifier to GateWay when making request. All requests by external systems are signed with RSA key (see chapter "Security").

All requests from external system must have their unique number "RequestId", which can be any combination of symbols allowed up to 64 bytes length.

## 3.2. Response

Every response has following XML structure:

```xml
<Response>
    <!-- Command response XML data -->
</Response>
```

When request fails or error occurs following XML structure is returned:

```xml
<Response>
    <Error Code="" Message=" "/>
</Response>
```

Error attributes description:

| Name | Type | Mandatory | Description |
|---|---|---|---|

| Code | Int | Yes | Error code |
|------|-----|-----|------------|
| Message | String(200) | Yes | Error message |

Error response example:
```xml
<Response>
    <Error Code="4" Message="Message is not xml"/>
</Response>
```

Possible error codes and their corresponding messages are listed in "Appendix A: Error codes".

# 4. Security

For security of data there are two levels used:

- First level – transport. Reachable by using secure HTTP protocol (HTTPS).
- Second level – signing of Data node with external system signature.

## 4.1. Signing of request

Every (external system), that has unique "Source" identifier, must create a pair of RSA keys (at least 1024 bits length) and provide public part. GateWay checks signature validity with public key.

Generated pairs of private and public key examples are provided in Appendix B.

Signature value (see "Basic XML structure") is calculated using agreed algorithm and encoded with BASE64 algorithm.

Encoding algorithm:
$$MAC(x1) := RSA(SHA\text{-}1(x1),d,n)$$

Where:
x1 – request "Data" node (for XML encoding rules see "4.2 "Data" XML format in request message and encoding for signature");
d – secret RSA exponent;
n - RSA module.

Example of signature calculation is provided in Appendix C.

## 4.2. "Data" XML format in request message and encoding for signature

The same valid XML document could be formatted (represented) as text and encoded (converted) to binary data (bytes) in different ways (due to whitespace, indentation, tabs vs spaces, new line formatting (unix/windows/mac), single vs double quotation marks, short vs extended empty nodes presentation and etc.). It is very important to consider possible differences and maintain the right format for properly signing request.

There is generally one main rule for "Data" XML fragment formatting in request message ("3.1 Request") and encoding it for signature:

> **XML "Data" fragment for signature should be encoded as UTF-8 bytes, and should be formatted exactly the same way (symbol-by-symbol), as it wil be included in request message.**

As it could be seen, exact XML format is not fixed – it is request composing party responsibility to choose one. However, it is highly adviseable to use some XML normalization techniques for "Data" fragment (consider to remove new line formatting and extra whitespace or indentation as a base technique of normalization).

# 5. Appendixes

## Appendix A: Error codes

| Code ("ErrorNo") | Message ("ErrorMsg") |
|---|---|
| 0 | … (Message directly could be shown to the end user, it will be localized by default partner language) |
| 1 | System error |
| 2 | Invalid signature |
| 3 | Unknown source |
| 4 | Message is not xml |
| 5 | Price is not number |
| 6 | Unknown regular price |
| 7 | Price is bigger that it is allowed, max allowed price is '...' |
| 8 | Price is smaller than it is allowed, min allowed price is '...' |
| 9 | Could not translate barcode |
| 10 | Product does not much with command |
| 11 | Sources does not much |
| 12 | Identificator length checking failed |
| 13 | Identificator comparison with date failed |
| 14 | Identificator comparison with symbols failed |
| 15 | Identificator LT personal id failed |
| 16 | Identificator LT company code failed |
| 17 | Identificator allowed symbols failed |
| 18 | Identificator LT payment book id failed |
| 19 | Identificator LuhnX failed |
| 20 | Identificator LuhnX failed in instruction |
| 21 | Identificator FI social security number failed |
| 22 | Identificator constant failed |
| 23 | Identificator regex failed |
| 24 | Identificator input not valid |
| 25 | Bill rows are not correct |
| 26 | Key not found for signature verification |
| 27 | Transaction was not found |
| 28 | Too much transactions found |
| 29 | Can`t find mentioned codes list for validation |
| 30 | Missing field PartnerTransactionId |
| 31 | Product could not be found |
| 32 | Invalid transition |
| 101 | Identificator ListOfCodes failed |
| 102 | Identificator Exclude ListOfCodes failed |
| 103 | Identificator control number 731 failed |
| 104 | Identificator control number failed |
| 105 | Transaction create and product kind does'nt much |

| 106 | Transaction create provided and estimated price does'nt much |
|-----|--------------------------------------------------------------|
|     |                                                              |

## Appendix B: RSA Key and certificate request example

Private key:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAlNQo2nLtDkgVshnE7o5LRYNaIkzM8ecTTkZCPujNX6WQeMMVPhBD7TVbc+lA
FEmQrrwIha2HsG+TvUCg10N7BRIhhG/4HFMb8soJJ2O1XqCGqRZ3lMmltJl5DGEa5lpfANlHSLIc
ypEmFXjguKxMeeNIyB9jrQSLUIXzkOjXqO/Ckf3FvtNFyjlS5p/ZF3D08pF3tv9/7kSsWtdeWPEF
SfOgMKJx5MiiMwlHG7fRxRY2dGqCGfHmQbHUsiFWVAuUkRJKNvb1xOU+cjxlIGBs/gzL8gCMUGCr
ckbQn+Ie6ZyDRdB8FeDINFUrC+3ohfb6MA1kQw5RgshxK4lDzd3W3QIDAQABAoIBACSclHg4sWxD
+2JWyIVO5+Fv0KqDbh9MEwm8auuC5ZLT7dhnVUBxS+R8rcSz6pmTB06U3xvTdn/G0vdJgMezx/2V
Wj10NfrxzvxJCztvYdynbLkH1974RBVLc+XBtZZZQOm6ibiuaDrEvjOmCtIjQKTzbqIcZq7XYVYr
vaE2Rmc+Q/pH2z2pxD4qdw8BfAAsx4ufOR8nO27juxk7IDVZvKh61EYgGPQh+Sb4Sug3X8nvMwnF
GzD/yRe0PxYesL3BbTc006R5EDAIm8z0Q0YOb05tFyKoI7Huuwm+v8xdZYYoxMb7+jjgLyXXnwUv
xu9V0787EBEVKCNGXk3GEHBuRoECgYEAxYB6ma4nUSsnVc7ReU4Kbx8qSnqO9O+BdEjd2zKcgItP
kn2e4nZFGWWfnPXmvlU2nS9GsNnwLTaze8bTexxaB4nPU65NPUFJEtvisBb9mEohY/OkiMYa3V8v
5iJCd/HY24/F01CFiuItIPBjoJ5ZgjKZg6nKanaqB+1pW4xxqm0CgYEAwOkONzAgsupbXJDDXrZd
eekElqqJ2Dpy51OVMrs2OK5jkre9a/x9CQ6vFtjEiBATecA7JOYFb+GHPixZBXrsGND8mx03CF4f
hfoh79C8FJ4gm4pkK4WpOS+syduTfaBMa9n6wtzBZ3aSCJIEGh/AOfM+2lcP/abuJVeWrZ9qGDEC
gYBzYaa+R3bivZJCWAwakRQZbRyY3FbOE2oOlE9AMqYplrmK/GapKK0ftDX2TdZa1Y+NbsQep4Pf
9jgPFM4R+zNBeiosWgT3xz84jEs30oaexJgTO5Cg8xNYxrZF/TfVuO2W/7xzwnPx+kahGq8pc7kZ
DhSPTc9aOS8GaJWHjVoOVQKBgFcIzuEXValjEtUwevTAUmwuNWPEjWyA8lZr32SEEnKAO2x2eX++
6qqrdVNZeVHp7t9/Wy/Kl6lilyH14DQYwWxw3YceJzYfQRmxQJ5yDmk+7nc+ePMp2q9yUwK/jkuf
htsh/YDtGFUVqc00E6W1rYR3umPH4+8M4WT3n4O16KgBAoGAfC1UP/fcz2tWrYsnlwwyW4KssU3u
SaYzkpUFlvZg9UrpcIskivYfXkzC2gZbKDx8GnnrkcVDBzp/MYT7c0j84R969RRaCYZvB/YCxsBt
iQ+D3i6nh12A8thAJXf0B6uIJ2VlMpABW76fZJDD858FwwgZZXb9woKL9IJ7PrsjDgs=
-----END RSA PRIVATE KEY-----

Public key:

-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAlNQo2nLtDkgVshnE7o5LRYNaIkzM8ecT
TkZCPujNX6WQeMMVPhBD7TVbc+lAFEmQrrwIha2HsG+TvUCg10N7BRIhhG/4HFMb8soJJ2O1XqCG
qRZ3lMmltJl5DGEa5lpfANlHSLIcypEmFXjguKxMeeNIyB9jrQSLUIXzkOjXqO/Ckf3FvtNFyjlS
5p/ZF3D08pF3tv9/7kSsWtdeWPEFSfOgMKJx5MiiMwlHG7fRxRY2dGqCGfHmQbHUsiFWVAuUkRJK
Nvb1xOU+cjxlIGBs/gzL8gCMUGCrckbQn+Ie6ZyDRdB8FeDINFUrC+3ohfb6MA1kQw5RgshxK4lD
zd3W3QIDAQAB
-----END PUBLIC KEY-----

## Appendix C:  Example of signing

Request data node:

```
<Data RequestId="1" Source="Source">
  <Transactions>
      <Create
          Product="Bill"
          Barcode="20707000010719"
          Price="50.00"
          ReceiptNo="45848"
          PartnerTransactionId="223"
          PrinterWidth="28"
          SalesPlace="CashRegister">
    </Create>
  </Transactions>
</Data>
```

Private key provided in Appendix B is used for signing.

"Signature" value:

bVzIKby/f+emdE3U4COtV1YDOYUGM2Ax12S1cHexzsaOcXQNzL/Hgi7y+/6DNlfDasvocyoqMY7pNAKbJzLj0
ai8VCTZLFzvpkqqMzWSidw3a7flZv67IfhWY10HoPtKKMvlm7EjE4rTglVvQi1o2ZirCuq/B0rZTjCEOHPNSn4V
+SjO8jmrdio3b86p/04OXmoqcPrKl9oF7zy5eZskGddvIlyqbtZRvjM2UsWaZ1ZbS+pVErzzukNoMvkkxXySNqK
eoD0pMjzfZxiOUT6kKOozgq1GpB9fJCm5NAtVx7AWkz70DZiiFhotTVFPGpAvXxaoHisUydD7YIXA2thlqQ==

## Appendix D: Example of client communication in C#

```csharp
// Generate .NET standart keys
// var provider = new RSACryptoServiceProvider();
// string privateKey = provider.ToXmlString();
// To generate public key, leave only Modulus and Exponent tags in privateKey data

var gateUrl = "http://localhost/GateWay/GateWay.ashx";

var xData = new XElement("Data",
        new XAttribute("RequestId", Guid.NewGuid()),
        new XAttribute("Source", "Source"),
        XElement.Parse("<Products><Info Product=\"123\" Barcode=\"\" /></Products>"));

var data =
"<RSAKeyValue><Modulus>6gu57Y7qWHFuPDWs9Hn2W3F6MxIIkL6Sfl4ruYIQ4m2myDcpIovZ7HRTyhDxH+L8Y6ulYNE8MfhChey9u2sJb90
KBZTUSCFh+iB5usZ4Af61f7MvJ6Aenad7sOymvJ0CRYGU/a1CZIyfZzNMvu0pQIznDnhYWKOWvkidM36p9Vc=</Modulus><Exponent>AQAB<
/Exponent><P>9a/OX5nVoA+6UKEnoGU4AWSbqQ99LbqWeVybbLRJQ3pUkgiztMO8JpJFuD4kjcChRFwRAfLSvxRvIL0QdLSIYQ==</P><Q>89
7TFMg5M7Q0BufPY7zfjJXQrPNZco+exALtFUXSY/gvFFFwQKcChJjWqN6PIafIKTpLlzRu+fhY4tsL9A94tw==</Q><DP>cAW7OYfxCcxiDYWN
herhAHGZMBqp3wHmvmtlSslhZm0Oxo9FMTTWQ1gKhhl/nUqpvXulHZhUU1/D08+hAzWVwQ==</DP><DQ>NccJXSfaFWDDqFIuFZxs35+Z6MmxC
+ZH6ImMpohDIPB10VE0I82wbHUXwR1zCsTpCs5kJO/XGSB+2WfcdvGCeQ==</DQ><InverseQ>eH8OeezcpYx9G4659rcb4jD7fo5DRmNrgP1G
14UE3RJ4ZOI5Id7T+DWlyGOxUxfiAPXybP0o65WbbVtihxC9Mg==</InverseQ><D>egN+/J/DADSyWWVKM3HZwzZXlIRUPtF7IbIKbX2nO8Qa
bMqFAWOTSrgibneCnTMCjRuQ7fZmwU5Br/wJtclJcy66JyrbIZCbUl2BVyzbCYuXm7K5Rcj5fS3EXDWwpYEfXtrhxVsYa3wr2k3Viia5f+ztbq
2CtAW1wtOcobpi1YE=</D></RSAKeyValue>";

var provider = new RSACryptoServiceProvider();
provider.FromXmlString(data);

var signature =
Convert.ToBase64String(provider.SignData(Encoding.UTF8.GetBytes(xData.ToString(SaveOptions.DisableFormatting))
, CryptoConfig.CreateFromName("sha1")));

var request = new XElement("Request", xData, new XElement("Signature",
signature)).ToString(SaveOptions.DisableFormatting);

string response = string.Empty;
HttpWebRequest httpRequest = (HttpWebRequest)WebRequest.Create(new Uri(gateUrl));

httpRequest.Method = "POST";
httpRequest.ContentType = "text; charset=utf-8";
httpRequest.KeepAlive = false;

httpRequest.ContentLength = request.Length;

using (Stream httpStream = httpRequest.GetRequestStream()) httpStream.Write(Encoding.UTF8.GetBytes(request),
0, request.Length);

using (HttpWebResponse httpResponse = (HttpWebResponse)httpRequest.GetResponse())
using (StreamReader sr = new StreamReader(httpResponse.GetResponseStream()))

// Response from gate
response = sr.ReadToEnd();
```