Parallels Business Automation 5.5

Public API Reference

Revision 1.2.14 (Aug 31, 2015)

Contents

CHAPTER 1

Preface

In This Chapter

Documentation Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

Typographical Conventions

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the **Resources** tab.
	Titles of chapters, sections, and subsections.	Refer to the **Managing Your Account** chapter.

Italics	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	Type a placeholder into the **From** field, e.g. *@@Order_Vendor_Account_BillingContact_Email@*
`Monospace`	The names of commands, files, and directories.	`payflow.verisign.com`
CAPITALS	Names of keys on the keyboard.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another.	CTRL+P, ALT+F4

Feedback

Parallels welcomes your feedback on how to make our documentation more helpful. You can submit feedback using the Documentation Feedback form on our website: http://www.parallels.com/support/usersdoc/?problemSubject=Feedback+on+Help+Page

Additionally, if you or your colleagues or customers have any feedback on the usability of Odin Service Automation, please send it to pa-feedback@parallels.com.

Please note that we do not respond to general questions and support requests submitted as feedback. If you submit a technical issue, it will be not responded. To receive customer support, please contact the Parallels support team: http://www.parallels.com/support/

Audience

This document can be used as a reference by customers who want to integrate Parallels Business Automation with their existing services, such as: online store, authorization service, provisioning systems and other external systems.

The reader is expected to be familiar with basic Parallels Business Automation concepts like: objects hierarchy, including accounts, users, subscriptions, and the like. The reader should also be familiar with resources accounting system used in Parallels Business Automation and understand the following terms: subscription, service plan, service template, and resource.

Scope

This document comprises reference and examples of use of Parallels Business Automation public API. Live usage scenarios applicable for migration, integration, and monitoring purposes are beyond the scope of this guide.

CHAPTER 2

Introduction

API methods are used by external systems for interaction with Parallels Business Automation (hereinafter - PBA). External systems implementation usually differs from PBA. So, the need of some agent for interaction comes up. The *XML-RPC* protocol is selected for this purpose. It is a specification and a set of implementations that allow systems running in different environments to make procedure calls over the Internet. A remote procedure call is performed using HTTP as the transport and XML as the encoding. For details on XML-RPC protocol, please, visit its official site http://www.xmlrpc.com.

Systems are interacts through built-in XML-RPC clients and servers. A client should be able to form requests according to XML-RPC specification and send these requests to a server of the other system. A server should be able to get requests from a client and process received requests.

Xmlrpcd is implemented on PBA side for interaction with the other systems. Xmlrpcd is a server application that provides infrastructure for invocation of any method of PBA servers available to user xmlrpcd runs on behalf of. Xmlrpcd is accessed through the following URL:

`http://<PBA Core server>:<port>/RPC2`,

where

- <PBA Core server> - is a host name or IP address of a server where PBA Core is installed;
- <port> - xmlrpcd port number listened by PBA Core server, by default it is 5224.

Note: both parameters are defined in the xproxy configuration.

Vendors may built their own XML-RPC clients to interact with PBA or use existing ones, for example *cURL*. The list of ready-to-use decisions can be found on XML-RPC official site.

CHAPTER 3

Using XMLRPC Daemon

To access PBA through *XML-RPC* protocol, special daemon program is used - *xmlrpcd*. Xmlrpcd is a daemon that provides infrastructure for invocation of any method of PBA servers available to user xmlrpcd runs on behalf of. Xmlrpcd also allows batch processing and provides facilities to control transaction flow.

Sending requests to xmlrpcd through cURL is described here as an example of interaction.

In This Chapter

Using Transaction Mechanism

PBA uses transaction mechanism to process requests. It works the following way: if request is processed successfully, made changes are committed; otherwise, made changes are rolled back.

XMLRPC daemon is implemented with transaction flow support. If you do not specify a transaction ID in a method a request, PBA creates new transaction, process the request commits or rolls back changes made by this request and closes the transaction. This behavior can be changed by switching off automatic commit for a particular request. The `AutoCommit` tag set to `No` is included into a request to prevent immediate commit. The following example shows fragment you need to include to a request to abolish automatic commit.

```
<member>
      <name>AutoCommit</name>
      <value>No</value>
</member>
```

PBA creates new transaction, processes the request, stores made changes and returns ID of created transaction. The transaction ID being included into a request prevents creation of new transaction and joins request processing to previously started transaction instead. The following example shows the fragment which allows joining to transaction #123.

```
<member>
      <name>TransactionID</name>
      <value><i4>123</i4></value>
</member>
```

The following methods are used for requests processing:

- `Execute` - the method request PBA to call particular method (`Method`) of specified PBA container (`Server`). Being called without transaction ID and with the `AutoCommit=No` option, returns ID of created transaction. Being called both with the transaction ID and the `AutoCommit=No` option, joins called methods processing in specified transaction.
- `CommitTransaction` - the method requests PBA to commit changes made by executed methods under particular transaction. The method requires transaction ID.
- `RollbackTransaction`.- the method requests PBA to roll back changes made by executed methods under particular transaction. The method requires transaction ID.

Note: the `TransactionID` parameter is always returned in XML response starting from PBA version 5.5

Example of Executing Request

To execute a request under the transaction #123 without committing changes, the following packet has to be sent.

```
<?xml version="1.0" ?>
<methodCall>
<methodName>Execute</methodName>
```

```
    <params>
     <param>
   <value>
      <struct>
          <member>
            <name>TransactionID</name>
            <value><i4>123</i4></value>
          </member>
          <member>
            <name>AutoCommit</name>
            <value>No</value>
          </member>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>Method_Name</value>
          </member>
          <member>
            <name>Params</name>
              <value>
                <array>
                  <data>
                    <value><i4>2</i4></value>
                    <value><i4>1</i4></value>
                    <value>string argument</value>
                    <value><double>123.456</double></value>
                  </data>
                </array>
              </value>
            </member>
          </struct>
         </value>
       </param>
     </params>
   </methodCall>
```

Example of Committing Transaction

To commit changes made under a transaction, for example, #123, the following packet has to be sent.

```
<?xml version="1.0" ?>
<methodCall>
 <methodName>CommitTransaction</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>TransactionID</name>
       <value><i4>123</i4></value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Example of Rolling Transaction Back

To roll back changes made under a transaction, for example, #123, the following packet has to be sent.

```
<?xml version="1.0" ?>
<methodCall>
 <methodName>RollbackTransaction</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>TransactionID</name>
       <value><i4>123</i4></value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Sending Request Through cURL

The cURL is a command line tool for getting or sending data using URL syntax. For details on cURL, its syntax and exit codes, please visit their official site http://curl.haxx.se/docs/manpage.html.

To post request to PBA we recommend the following command:

```
curl --connect-timeout 10 -d@'
<xml request>
' -H 'Content-type:text/xml' http://<PBA Core server>:<Port>/RPC2
```

where

- <xml request> - PBA API method call formed according to XML-RPC specification;
- <PBA Core server> - is a host name or IP address of a server where PBA Core is installed;
- <port> - xmlrpcd port number listened by PBA Core server, by default it is 5224.

Executing Request under User Credentials

By default, all calls from xmlrpcd to PBA servers are executed under credentials of xmlrpcd. This default behavior can be easily overwritten by tags `Username` and `Password`. These tags must be used together. `Username` is string identifier of the user, his login name. That is if you want to call BM under the user `admin` having password "1q2w3e" use the following example:

```
<member>
      <name>Username</name>
      <value>admin</value>
</member>
<member>
      <name>Password</name>
      <value>1q2w3e</value>
</member>
```

Specifying Locale

By default all open API methods return string values defined in the `en` locale as well as create new objects like accounts and users with default English language. Such behaviour can be changed by adding to the method request an additional parameter `Lang` and specifying needed locale as its value, for example:

```
<member>
  <name>Lang</name>
  <value><string>de_DE</string></value>
</member>
```

The `Lang` parameter should be added to the request after the `Method` parameter. For example, to get a service plan details in German send the following request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanDetailsGet_API</value>
     </member>
     <member>
      <name>Lang</name>
      <value><string>de_DE</string></value>
     </member>
     <member>
      <name>Params</name>
      <value>
```

```
       <array>
        <data>
   <!-- PlanID -->
         <value><i4>3</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Using Simple Method Invocation

Method MyFirstMethod(int, int, char*, double) is called from Object BM_Object of Container BM_Container.

```
<?xml version="1.0" ?>
<methodCall>
<methodName>Execute</methodName>
 <params>
  <param>
    <value>
      <struct>
        <member>
             <name>Container</name>
              <value>BM_Container</value>
        </member>
        <member>
         <name>Object</name>
         <value>BM_Object</value>
        </member>
        <member>
        <name>Method</name>
          <value>MyFirstMethod</value>
        </member>
        <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <value><i4>2</i4></value>
           <value><i4>1</i4></value>
           <value>string argument</value>
           <value><double>123.456</double></value>
          </data>
             </array>
           </value>
          </member>
         </struct>
        </value>
    </param>
   </params>
</methodCall>
```

In fact, each Server is a Container of Objects which in turn contains methods, but this flexibility is rarely used. More often server contains only one Object. Naming convention is the following:

- Server name is an uppercase string of letters and underscores (BM, BLACK_EMAIL, PEMGATE) this name is used only for convenience - in fact it's redundant;
- Container name is a Server name, followed by "_Container" suffix (BM_Container, BLACK_EMAIL_Container, PEMGATE_Container);
- Object name is an uppercase string of letters and underscores followed by "_Object" suffix. Every Container contains at least one Object. In the case of single Object this Object has name constructed from Server name and "_Object" suffix (BM_Object, BLACK_EMAIL_Object, PEMGATE_Object).

Using these naming rules we can substitute pair of Container-Object by short Server name. For example BM_Container.BM_Object can be substituted by BM. In this case the previous call can be substituted by the following:

```
<?xml version="1.0" ?>
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
             <name>Server</name>
             <value>BM</value>
         </member>
      <member>
             <name>Method</name>
             <value>MyFirstMethod</value>
         </member>
         <member>
             <name>Params</name>
                    <value>
                     <array>
                      <data>
                           <value><i4>2</i4></value>
                           <value><i4>1</i4></value>
                           <value>string argument</value>
                           <value><double>123.456</double></value>
                       </data>
                      </array>
                     </value>
                    </member>
                 </struct>
                </value>
               </param>
              </params>
             </methodCall>
```

This simplest example shows the way of calling method in new transaction. All changes made by this method are immediately committed.

Depending on the type of the result, returning by the method, three types of response packets can be returned by xmlrpcd:

- Status message (ErrorMessage in Stellart terms);
- Structure (ItemResult in Stellart terms);
- Array (ListResult is Stellart terms).

Example of Status Result

Note: ID of newly started and not committed transaction is returned together with status message.

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <member>
                        <name>Status</name>
                        <value>Everything is OK</value>
                      </member>
                    </struct>
                  </value>
                </data>
              </array>
            </value>
          </member>
          <member>
            <name>TransactionID</name>
            <value><i4>123</i4></value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Example of Structure Result (ItemResult)

Structure consists of 5 slots - int, char*, double, int, char*

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value><i4>1</i4></value>
                        <value>second slot (string)</value>
                        <value><double>1.23</double></value>
                        <value><i4>1</i4></value>
                        <value>Another string slot</value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Example of Array Result (ListResult)

Array consists of 6 rows and 2 columns.

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value>
                          <array>
                            <data>
                              <value><i4>1</i4></value>
                              <value>Disk Space</value>
                              <value>MB</value>
                              <value><i4>200</i4></value>
                              <value><double>0.000000</double></value>
                              <value><i4>0</i4></value>
                            </data>
                          </array>
                        </value>
                        <value>
                          <array>
                            <data>
                              <value><i4>2</i4></value>
                              <value>Traffic</value>
                              <value>GB</value>
                              <value><i4>20</i4></value>
                              <value><double>1.000000</double></value>
                              <value><i4>0</i4></value>
                            </data>
                          </array>
                        </value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Example of Batch Request

Request

```
<?xml version="1.0" ?>
<methodCall>
<methodName>Execute</methodName>
<params>
  <param>
    <value>
      <array>
        <data>
          <value>
            <struct>
              <member>
                <name>TransactionID</name>
                <value><i4>123</i4></value>
              </member>
              <member>
                <name>AutoCommit</name>
                <value>No</value>
              </member>
              <member>
                <name>Server</name>
                <value>BM</value>
              </member>
              <member>
                <name>Method</name>
                <value>MyFirstMethod</value>
              </member>
              <member>
                <name>Params</name>
                <value>
                  <array>
                    <data>
                      <value><i4>2</i4></value>
                      <value><i4>1</i4></value>
                      <value>string argument</value>
                      <value><double>123.456</double></value>
                    </data>
                  </array>
                </value>
              </member>
            </struct>
          </value>
          <value>
            <struct>
              <member>
                <name>Server</name>
                <value>BM</value>
              </member>
              <member>
                <name>Method</name>
                <value>MySecondMethod</value>
              </member>
              <member>
                <name>Params</name>
                <value>
                  <array>
```

```
                    <data>
                      <value><i4>77</i4></value>
                    </data>
                  </array>
                </value>
              </member>
            </struct>
          </value>
        </data>
      </array>
    </value>
  </param>
</params>
</methodCall>
```

Response

```
<?xml version="1.0" ?>
<<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <struct>
                      <member>
                        <name>Status</name>
                        <value>Everything is OK</value>
                      </member>
                    </struct>
                  </value>
                  <value>
                    <array>
                      <data>
                        <value><i4>1</i4></value>
                        <value>second slot (string)</value>
                        <value><double>1.23</double></value>
                        <value><i4>1</i4></value>
                        <value>Another string slot</value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

Error Messages

If method returns an error, the error message is contained in the fault string; which is Base64 encoded. The example of method response containing the fault string is provided below:

```
< ?xml version="1.0"?>
       < methodResponse>
<fault>
           <value>
  <struct>
       <member>
        <name>faultCode</name>
        <value>
         <i4>-1</i4>
        </value>
       </member>
       <member>
        <name>faultString</name>
         <value>
          <string>
0JLQstC10LTRkdC90L3Ri9C5INCb0L7Qs9C40L0g0Y/QstC70Y/QtdGC0YHRjyDQvdC10LrQvtGA0YDQtdC60YL
QvdGL0LwuINCf0L7QttCw0LvRg9C50YHRgtCwLCDRg9C00L7RgdGC0L7QstC10YDRjNGC0LXRgdGMLCDRh9GC0L
4g0LTQu9C40L3QsCDQm9C+0LPQuNC90LAg0YHQvtGB0YLQsNCy0LvRj9C10YIg0L7RgiA1INC00L4gMjAg0YHQu
NC80LLQvtC70L7QsiDQuCDQvdC1INGB0L7QtNC10YDQttC40YIg0YHQv9C10YbQuNCw0LvRjNC90YvQtSDQuNC7
0Lgg0L3QsNGG0LjQvtC90LDQu9GM0L3Ri9C1INGB0LjQvNCy0L7Qu9GLLiBMb2dpbiBOYW1lIGlzIGluY29ycmV
jdC4gUGxlYXNlIG1ha2Ugc3VyZSB0aGF0IHlvdXIgTG9naW4gTmFtZSBpcyA1IHRvIDIwIGNoYXJhY3RlcnMgbG
9uZyBhbmQgZG9lcyBub3QgY29udGFpbiBzcGVjaWFsIG9yIG5hdGlvbmFsIGNoYXJhY3RlcnMuIExvZ2luIG11c
3QgYmUgbW9yZSB0aGFuIDUgY2hhcmFjdGVyKHMpIGFuZCBsZXNzIHRoYW4gMjAgY2hhcmFjdGVyKHMpIGxvbmcu
CgoKCgoKCgo=
          </string>
     </value>
    </member>
   </struct>
  </value>
           </fault>
</methodResponse>
```

CHAPTER 4

XMLRPC Advanced Configuration

PBA XMLRPC API Scalability

The scalability of PBA API is can be significantly improved by increasing the number of threads for xmlrpcd service, and will be only bounded only by CPU quantity on the PBA node, the scalability is almost linear up to 16 CPUs.

Enable authentication for external systems that access PBA
PBA may require client authorization through XMLRPCD. If authentication is enabled, it is possible to allow access to PBA from outside network provided that PBA user login and password are passed. In this case, you need to configure the second xmlrpcd instance to run, specifying a second port to which the xmlrpcd service would listen to. Additionally, you can configure it to use SSL.

To configure xmlrpcd follow the directions below.

1 Create a copy of the .xmlrpcd.conf configuration file, for example:

```
cp
$PBA_ROOT/etc/ssm.conf.d/.xmlrpcd.conf $PBA_ROOT/etc/ssm.conf.d/
xmlrpcd_auth.conf
```

2 Edit the file `$PBA_ROOT/etc/ssm.conf.d/xmlrpcd_auth.conf` as follows:

```
[environment]
ATMName = XMLRPC_Auth_Container
AuthorizationRequired = 1
[options]
bin = xmlrpcd$(_eext)
bindir = $(st_prefix)/$(_bindir_name)
startdep = atm
summary = Stellart XML RPC AUTH Server
arguments = $(HOST_IP):5225
```

3 In case of using SSL, set the respective parameters:

```
SslPrivateKey = %path_to_private_key%
SslCertificate = %path_to_certificate%
```

4 For scalability, specify the desired number of threads at the `[environment]` section with the `Threads` parameter, for example, `Threads = 5`

5 After saving the file, run the `configure.pl` configuration script. It will update the start-up parameters of the `global.conf` file:

- For Linux: `$PBA_ROOT/tools/configure.pl`
- For Windows: `$PBA_ROOT\perl\bin\perl.exe tools\configure.pl`

6 Restart xmlrpcd:

- If you have changed the number of threads:
  - For Linux: `/etc/init.d/xmlrpcd restart`
  - For Windows: restart the Stellart XML RPC Server service
- If you have configured a second xmlrpc instance:
  - For Linux: `/etc/init.d/xmlrpcd_auth restart`

  For Windows: restart the Stellart XML RPC AUTH Server service After restart, a separate instance of the xmlrpc service will be launched.

CHAPTER 5

Usage Scenarios

This chapter provides you with a number of real life API methods application scenarios. These scenarios will illustrate how and what methods you should use to match your needs. The knowledge will help you to solve several typical tasks, as well as efficiently solve ones that may arise before you in future.

In This Chapter

Entities Migration to PBA

The section provides information on how to use PBA API methods for migrating business entities from alike systems to PBA.

Account Migration

Standalone PBA

In PBA each entity which can charge for services and/or be billed for services is provided an account. Each account is required to have at least one user for logging into system and performing some actions. When account is being added through PBA CP or online store a user is created for the account simultaneously. Actually, account and user creation are separate operations, not related to each other.

The following sequence of API methods is used for account creation:

1. `AccountAdd_API` (on page 43) - creates an account in PBA.
2. `UserAdd_API` (on page 633) - adds a user for specified account.

PBA integrated with POA

In case of integration an account must be present in both systems. By default, when account is created in PBA, an event of `Account Created` type is occurred. This event is processed by `AddAccountToPEM` event handler that triggers respective POA API method. The handler has the following restriction: account to be added to POA must have at least one user.

To work around this restriction the handler settings should be changed as following:

- **Max attempts** = *2* or more (1 by default);
- **Repeat interval (seconds)** = *several seconds* (1 by default).

Due to this modification of the handler settings, the handler execution fails on first attempt, but because of extended repeat interval there is a time to add user to the account and the second handler attempt succeeds.

Hosting Subscription Migration

Standalone PBA

Make sure, that service plan and subscription period you are going to subscribe account to exist in PBA. Hosting subscriptions are based on DUMMYGATE, so `SubscriptionAdd_API` (on page 585) method should be used.

PBA integrated with POA

In case of integration the subscription must be present in both systems and be based on PEMGATE. API methods of both PBA and POA systems are involved. For details on POA API methods, refer to the POA respective guide.

The following sequence of API methods is used for hosting subscription creation:

1. `AddPEMSubscription_API` (on page 103) - creates hosting subscription in PBA.
2. `pem.activateSubscription` - creates hosting subscription in POA.

Domain Subscription Migration

Standalone PBA

Preconditions:

- Registrar plug-in and domain zone are configured as required;
- Service plan and subscription period you are going to subscribe account to exist in PBA.

Domain subscriptions are added using `DomainSubscrAdd_API` (on page 235) method.

PBA integrated with POA

In case of integration the subscription must be present in both systems. Also, a DNS hosting subscription should be created in POA for the domain subscription. API methods of both PBA and POA systems are involved. For details on POA API methods, refer to the POA respective guide.

The following sequence of API methods is used for domain subscription creation:

1. `DomainSubscrAdd_API` (on page 235) - creates domain subscription in PBA;
2. `pem.addSubscription` - creates default DNS hosting subscription in POA with the same ID as created domain subscription got in PBA (step 1);
3. `pem.addDomain` - assignes DNS hosting to POA default DNS subscription. Domain is registered in POA and POA, method returns value for service parameter `domainID`;

Important: To avoid possible issues, use `DomainExtDataAdd_API` (on page 213) to add to the domain all necessary data, which has been already added to it before.

Hosting + Domain Subscriptions Migration

Standalone PBA

Preconditions:

- Registrar plug-in and domain zone are configured as required;
- Service plans (domain and hosting) and subscription periods you are going to subscribe account to exist in PBA;
- Domain service plan is assigned to hosting one as up-sale.

The following sequence of API methods is used for hosting with domain as up-sale subscriptions creation:

1. `SubscriptionAdd_API` (on page 585) - creates hosting subscription based on DUMMYGATE. The method returns added subscription ID;

2 `DomainSubscrAdd_API` (on page 235) - creates domain subscription in PBA, hosting subscription (step 1) is defined as parent one in method call.

PBA integrated with POA

In case of integration subscriptions must be present in both systems. Also, a DNS hosting subscription should be created in POA for the domain subscription. API methods of both PBA and POA systems are involved. For details on POA API methods, refer to the POA respective guide.

The following sequence of API methods is used for hosting with domain as up-sale subscriptions creation:

1 `AddPEMSubscription_API` (on page 103) - creates hosting subscription in PBA;

2 `pem.activateSubscription` - creates hosting subscription in POA;

3 `DomainSubscrAdd_API` (on page 235) - creates domain subscription in PBA, hosting subscription (step 1) is defined as parent one in method call;

4 `pem.addSubscription` - creates default DNS hosting subscription in POA with the same ID as created domain subscription got in PBA (step 1);

5 `pem.addDomain` - assigns DNS hosting to POA default DNS subscription. Domain is registered in POA and POA, method returns value for service parameter `domainID`;

6 `pem.bindServicesToDomain` - assigns hosting subscription to domain using `domainID` returned in step 5;

7 `AddPEMDomainForSubscription_API` (on page 112) - completes necessary settings configuration at PBA side.

Important: Avoid Sensitive Data Logging

Any call of any API method in PBA is logged.

The data passed to API methods may include user passwords, credit card numbers, credit cards CVV, any other sensitive data that must be stored in a secure storage and never be logged.

To avoid logging of sensitive data during API calls toPBA, the "XXX" prefix should be added to the relevant arguments.

The prefix is case-sensitive: XXX should be in upper case only.

For example, credit card number as 4111111111111111 should be passed as XXX4111111111111111. As a result, credit card numbers will be passed to PBA as plain text with XXX prefix. All the data passed in such a way will be not logged. On PBA side, the data with XXX prefix is recognized, the prefix is dropped, then the further data processing is performed with clear data, without prefix.

The following API methods pass the sensitive data:

- CreditCard2AccountAdd_API (on page 180)
- UserAdd_API (on page 633)
- UserPasswdUpdate_API (on page 644)
- UserWithIDAdd_API (on page 661)
- UserWithIDAddWithRole_API (on page 665)
- PlaceOrderAndAuthorize_API (on page 391)

CHAPTER 6

API Methods

This section covers available PBA Open API methods. XML requests formed according to XML-RPC specification are provided for each method as an example.

Important: XML entities should be encoded properly according to the XML format; that is, one should use &amp; instead of & and so on.

AccountAdd_API

	Parameters	
Description	**Passed**	**Returned**
Method creates new account in PBA with ID automatically assigned by system of with one specified in call. Method has 1 optional parameter that can be added to the begging of the call (new account ID) and 3 optional parameters that can be added to the end of the call, related to taxation. Syntax is given with all parameters available. There are two types of account in PBA: person and company. Company account needs company name, address and three types of contacts (administrative, billing and technical) to be passed. Personal account needs address and personal details only. Example below is given for company account type.	62 or more	1

Syntax

```
ItemResult BM::AccountAdd_API(
  Str AccountID;
  Int VendorAccountID;
  Int VType;
  Str(80) CompanyName;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(1024) PostalAddress;
  Str(30) AdminFName;
  Str(30) AdminMName;
  Str(30) AdminLName;
  Str(100) AdminEmail;
  Str(4) AdminPhCountryCode;
  Str(10) AdminPhAreaCode;
  Str(20) AdminPhNumber;
  Str(10) AdminPhExtention;
  Str(4) AdminFaxCountryCode;
  Str(10) AdminFaxAreaCode;
  Str(20) AdminFaxNumber;
  Str(10) AdminFaxExtention;
  Str(30) BillFName;
  Str(30) BillMName;
  Str(30) BillLName;
  Str(100) BillEmail;
  Str(4) BillPhCountryCode;
  Str(10) BillPhAreaCode;
  Str(20) BillPhNumber;
  Str(10) BillPhExtention;
  Str(4) BillFaxCountryCode;
  Str(10) BillFaxAreaCode;
  Str(20) BillFaxNumber;
  Str(10) BillFaxExtention;
  Str(30) TechFName;
  Str(30) TechMName;
  Str(30) TechLName;
  Str(100) TechEmail;
  Str(4) TechPhCountryCode;
  Str(10) TechPhAreaCode;
  Str(20) TechPhNumber;
  Str(10) TechPhExtention;
  Str(4) TechFaxCountryCode;
  Str(10) TechFaxAreaCode;
  Str(20) TechFaxNumber;
  Str(10) TechFaxExtention;
  Str(30) PersFName;
  Str(30) PersMName;
  Str(30) PersLName;
  Int Birthday;
  Str(1024) Passport;
  Str(100) PersEmail;
  Str(4) PersPhCountryCode;
  Str(10) PersPhAreaCode;
  Str(20) PersPhNumber;
  Str(10) PersPhExtention;
  Str(4) PersFaxCountryCode;
  Str(10) PersFaxAreaCode;
  Str(20) PersFaxNumber;
  Str(10) PersFaxExtention;
  Int TaxStatus;
  Str(10) TaxZoneID;
  Str(20) TaxRegID;
  Int TaxRegIDStatus;
  Str(20) Originator.
)
returns
  Int AccountID - created account ID.
```

Special Notes

Note: Company specific parameters should be passed as empty/zero, if personal account is added and vice versa.

The following parameters are relevant for both types of account (company and person).

- **Str** AccountID - ID of the account to be created. The parameter is optional and should be passed as following: `AccountID=`<*account_ID*>, for example, `<value>AccountID=100025689</value>`;
- **Int** VendorAccountID - vendor account ID (provider or reseller);
- **Int** VType - account role: 2 - Reseller, 3 - Customer;
- **Str(80)** CompanyName - account name. It can be company name or person first and last names;
- **Str(80)** Address1 - account address (line 1/2);
- **Str(80)** Address2 - account address (line 2/2);
- **Str(40)** City - account address: city;
- **Str(80)** State - account address: state;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(1024)** PostalAddress - account address in free form;

The following parameters are relevant for company account type. For person account type should be passed empty.

- **Str(30)** AdminFName - account administrative contact information: first name;
- **Str(30)** AdminMName - account administrative contact information: middle name;
- **Str(30)** AdminLName - account administrative contact information: last name;
- **Str(100)** AdminEmail - account administrative contact information: e-mail address;
- **Str(4)** AdminPhCountryCode - account administrative contact information: phone country code;
- **Str(10)** AdminPhAreaCode - account administrative contact information: phone area code;
- **Str(20)** AdminPhNumber - account administrative contact information: phone number;
- **Str(10)** AdminPhExtention - account administrative contact information: phone extension;

- **Str(4)** AdminFaxCountryCode - account administrative contact information: fax country code;
- **Str(10)** AdminFaxAreaCode - account administrative contact information: fax area code;
- **Str(20)** AdminFaxNumber - account administrative contact information: fax number;
- **Str(10)** AdminFaxExtention - account administrative contact information: fax extension;
- **Str(30)** BillFName - account billing contact information: first name;
- **Str(30)** BillMName - account billing contact information: middle name;
- **Str(30)** BillLName - account billing contact information: last name;
- **Str(100)** BillEmail - account billing contact information: e-mail address;
- **Str(4)** BillPhCountryCode - account billing contact information: phone country code;
- **Str(10)** BillPhAreaCode - account billing contact information: phone area code;
- **Str(20)** BillPhNumber - account billing contact information: phone number;
- **Str(10)** BillPhExtention - account billing contact information: phone extension;
- **Str(4)** BillFaxCountryCode - account billing contact information: fax country code;
- **Str(10)** BillFaxAreaCode - account billing contact information: fax area code;
- **Str(20)** BillFaxNumber - account billing contact information: fax number;
- **Str(10)** BillFaxExtention - account billing contact information: fax extension;
- **Str(30)** TechFName - account technical contact information: first name;
- **Str(30)** TechMName - account technical contact information: middle name;
- **Str(30)** TechLName - account technical contact information: last name;
- **Str(100)** TechEmail - account technical contact information: e-mail address;
- **Str(4)** TechPhCountryCode - account technical contact information: phone country code;
- **Str(10)** TechPhAreaCode - account technical contact information: phone area Code;
- **Str(20)** TechPhNumber - account technical contact information: phone number;
- **Str(10)** TechPhExtention - account technical contact information: phone extension;
- **Str(4)** TechFaxCountryCode - account technical contact information: fax country code;
- **Str(10)** TechFaxAreaCode - Account Tech contact information: Fax Area Code;
- **Str(20)** TechFaxNumber - account technical contact information: fax number;
- **Str(10)** TechFaxExtention - account technical contact information: fax extension;

The following parameters are relevant for person account type. For company account type should be passed empty.

- **Str(30)** PersFName - account personal contact information: first name;
- **Str(30)** PersMName - account personal contact information: middle name;

- **Str(30)** PersLName - account personal contact information: last name;
- **Int** Birthday - account personal contact information: birthday in Unix date format;
- **Str(1024)** Passport - account personal contact information: passport details;
- **Str(100)** PersEmail - account personal contact information: e-mail address;
- **Str(4)** PersPhCountryCode - account personal contact information: phone country code;
- **Str(10)** PersPhAreaCode - account personal contact information: phone area code;
- **Str(20)** PersPhNumber - account personal contact information: phone number;
- **Str(10)** PersPhExtention - account personal contact information: phone extension;
- **Str(4)** PersFaxCountryCode - account personal contact information: fax country code;
- **Str(10)** PersFaxAreaCode - account personal contact information: fax area code;
- **Str(20)** PersFaxNumber - account personal contact information: fax number;
- **Str(10)** PersFaxExtention - account personal contact information: fax extension;

The following parameters are relevant for both types of account (company and person).

- **Int** TaxStatus - account type: 1 - Person, 2 - Company;
- **Str(10)** TaxZoneID - ID of tax zone to regulate account taxation. The list of tax zones available can be accessed from the **System** > **Settings** > **Taxation** > **Tax Zones** submenu of the Navigation tree;

The following parameters are optional for both types of account (company and person).

- **Str(20)** TaxRegID - customer tax registration ID;
- **Int** TaxRegIDStatus - defines whether tax registration ID is checked: 0 - not verified, 1 - verified;
- **Str(20)** Originator - identifies the initiator of the account creation. Depending on the passed value, the account is created either in PBA only or both in PBA and POA:
  - If `PEM` is passed as the Originator value, then the account is created only in PBA. The *Account Created* event is posted; the originator value is passed to the event and becomes the event parameter. Use case: the account already exists in POA, so it is needed to create it only at PBA side.
  - If other non-empty value is passed as the Originator value, then the account is created both in PBA and in POA. The *Account Created* event is posted; the originator value is passed to the event and becomes the event parameter. Use case: if you migrate accounts to PBA and POA using API you can pass, for example, "API Migration <*date*>". Then you can track added accounts in the event log by this originator.

  Note: The event settings can be viewed on the **System** > **Settings** > **Events** screen.

  - If the Originator is not passed (NULL) then the *Account Created* event is not posted. The account is created only in PBA. Use case: it is needed to create the account on the stand-alone PBA installation.

By default, the PBA account creation wizard posts the Account Created event and passes the non-empty value (by default, `BM`) to the event. The event handler calls pem.AddAccountToPEM method of the PEMGATE and the account is created both in PBA and POA.

> **Important**: If you pass the Originator, the parameters TaxRegID and TaxRegIDStatus should be also passed, at least with the empty values.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- VType -->
         <value><i4>3</i4></value>
         <!-- CompanyName -->
         <value>JV SkyWings</value>
         <!-- Address1 -->
         <value>Sunrise Valley Drive</value>
         <!-- Address2 -->
         <value>Suite 600</value>
         <!-- City -->
         <value>Herndon</value>
         <!-- State -->
         <value>VA</value>
         <!-- Zip -->
         <value>20171</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PostalAddress -->
         <value>13755 Sunrise Valley Drive, USA</value>
         <!-- AdminFName -->
         <value>John</value>
         <!-- AdminMName -->
```

```
<value>M.</value>
<!-- AdminLName -->
<value>Smith</value>
<!-- AdminEmail -->
<value>jsmith@skywings.com</value>
<!-- AdminPhCountryCode -->
<value>1</value>
<!-- AdminPhAreaCode -->
<value>703</value>
<!-- AdminPhNumber -->
<value>8155670</value>
<!-- AdminPhExtention -->
<value>111</value>
<!-- AdminFaxCountryCode -->
<value>1</value>
<!-- AdminFaxAreaCode -->
<value>703</value>
<!-- AdminFaxNumber -->
<value>8155670</value>
<!-- AdminFaxExtention -->
<value>112</value>
<!-- BillFName -->
<value>John</value>
<!-- BillMName -->
<value>M.</value>
<!-- BillLName -->
<value>Smith</value>
<!-- BillEmail -->
<value>jsmith@skywings.com</value>
<!-- BillPhCountryCode -->
<value>1</value>
<!-- BillPhAreaCode -->
<value>703</value>
<!-- BillPhNumber -->
<value>8155670</value>
<!-- BillPhExtention -->
<value>111</value>
<!-- BillFaxCountryCode -->
<value>1</value>
<!-- BillFaxAreaCode -->
<value>703</value>
<!-- BillFaxNumber -->
<value>8155670</value>
<!-- BillFaxExtention -->
<value>112</value>
<!-- TechFName -->
<value>John</value>
<!-- TechMName -->
<value>M.</value>
<!-- TechLName -->
<value>Smith</value>
<!-- TechEmail -->
<value>jsmith@skywings.com</value>
<!-- TechPhCountryCode -->
<value>1</value>
<!-- TechPhAreaCode -->
<value>703</value>
<!-- TechPhNumber -->
```

```
          <value>8155670</value>
          <!-- TechPhExtention -->
          <value>111</value>
          <!-- TechFaxCountryCode -->
          <value>1</value>
          <!-- TechFaxAreaCode -->
          <value>703</value>
          <!-- TechFaxNumber -->
          <value>8155670</value>
          <!-- TechFaxExtention -->
          <value>112</value>
          <!-- PersFName -->
          <value></value>
          <!-- PersMName -->
          <value></value>
          <!-- PersLName -->
          <value></value>
          <!-- Birthday -->
          <value><i4></i4></value>
          <!-- Passport -->
          <value></value>
          <!-- PersEmail -->
          <value></value>
          <!-- PersPhCountryCode -->
          <value></value>
          <!-- PersPhAreaCode -->
          <value></value>
          <!-- PersPhNumber -->
          <value></value>
          <!-- PersPhExtention -->
          <value></value>
          <!-- PersFaxCountryCode -->
          <value></value>
          <!-- PersFaxAreaCode -->
          <value></value>
          <!-- PersFaxNumber -->
          <value></value>
          <!-- PersFaxExtention -->
          <value></value>
          <!-- TaxStatus -->
          <value><i4>2</i4></value>
          <!-- TaxZoneID -->
          <value>Default</value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added Account ID -->
              <value><i4>1000017</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountBalanceAdjust_API

Description	Parameters	
	Passed	Returned
Method adjusts account balance for specified amount and creates appropriate adjustment document. **Note:** you have to manually specify tax category, sales branch and sales person for an adjustment document created with the method.	2	1

Syntax

```
ErrorMessage BM::AccountBalanceAdjust_API(
  Int AccountID;
  Double BalanceDif.
)
returns
 Str ErrorMessage - message after account balance has been updated: "Account
balance is successfully updated.".
```

Special Notes

- **Int** AccountID - account ID;
- **Double** Balancedif - adjustment amount. If the sum is positive credit memo on specified sum is created, otherwise debit memo.

Note: account balance is shown differently in Customer Control Panel and Provider Control Panel (i.e. it will be positive when viewing in from CCP and negative in PCP, and vice versa).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountBalanceAdjust_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
         <!-- BalanceDif -->
         <value><double>123</double></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Account balance is successfully updated
               </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountBalanceUpdate_API

Description	Parameters	
	Passed	Returned
Method assigns new value to account **Balance** parameter and creates appropriate adjustment document for `NewBalance - CurrentBalance` amount.	2	1

Syntax

```
ErrorMessage BM::AccountBalanceUpdate_API(
  Int AccountID;
  Double NewBalance.
)
returns
  Str ErrorMessage - message on Balance parameter value change: "Account balance is
successfully assigned.".
```

Special Notes

- **Int** AccountID - account ID;
- **Double** NewBalance - account new balance sum, passed in "00.00" format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountBalanceUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
         <!-- NewBalance -->
         <value><double>124</double></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
               Account balance is successfully updated
               </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountBalanceUpdateTimeUpdate_API

	Parameters	
Description	**Passed**	**Returned**
Method assigns new value to account **Balance Last Update Time** parameter.	2	1

Syntax

```
ErrorMessage BM::AccountBalanceUpdateTimeUpdate_API(
  Int AccountID;
  Int UpdateTime.
)
returns
 Str ErrorMessage - message on Balance Last Update Time parameter value change:
"Account balance update time is successfully assigned.".
```

Special Notes

- **Int** AccountID - account ID;
- **Int** UpdateTime - date and time you want to assign in Unix date format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountBalanceUpdateTimeUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
         <!-- BalanceUpdateTime -->
         <value><i4>1161373724</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Account balance update time is successfully assigned.
               </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountDetailsGet_API

Description	Parameters	
	Passed	**Returned**
This method returns the details on the specified account. Which contact information is returned (a company (or person) name, email, phone number, and fax) depends on the account type: • If the account type is **Company**, the administrative contact information is returned. • If the account type is **Personal**, the personal contact information is returned.	1	26

The syntax below is given for company accounts.

Syntax

```
ItemResult BM::AccountDetailsGet_API(
Int AccountID.
)
```

```
returns
Int AccountID;
Int VendorAccountID;
Str CompanyName;
Str Address1;
Str Address2;
Str City;
Str State;
Str Zip;
Str CountryID;
Str PostalAddress;
Str accFName;
Str accMName;
Str accLName;
Str accEmail;
Str accPhCountryCode;
Str accPhAreaCode;
Str accPhNumber;
Str accPhExtention;
Str accFaxCountryCode;
Str accFaxAreaCode;
Str accFaxNumber;
Str accFaxExtention;
Int CreationDate;
Int TaxStatus;
Int AStatus;
Int FullyRegistred.
```

Parameters description

Input parameter:

- **Int** AccountID - ID of the account for which you want to get the details.

Output parameters:

- **Int** AccountID - The account identifier.
- **Int** VendorAccountID - The identifier of the vendor (provider or reseller) account to which the specified customer account belongs.
- **Str** CompanyName - The company name (for an account of the **Company** type) or the first and last names (for a **Personal** account).
- **Str** Address1 - The account owner's address 1 (line 1/2).
- **Str** Address2 - The account owner's address 2 (line 2/2).
- **Str** City - The account owner's home city.
- **Str** State - The account owner's home state or province.
- **Str** Zip - The account owner's postal code.
- **Str** CountryID - The two-letter code of the account owner's home country.
- **Str** PostalAddress - The account owner's postal address written in the free form.
- **Str** accFName - The first name of the account owner's administrative contact.

- **Str** accMName - The middle name of the account owner's administrative contact.
- **Str** accLName - The last name of the account owner's administrative contact.
- **Str** accEmail - The email of the account owner's administrative contact.
- **Str**accPhCountryCode - The country code of the administrative contact phone number.
- **Str** accPhAreaCode - The area code of the administrative contact phone number.
- **Str** accPhNumber - The administrative contact phone number.
- **Str** accPhExtention - The extension of the administrative contact phone number.
- **Str** accFaxCountryCode - The country code of the administrative contact fax number.
- **Str** accFaxAreaCode - The area code of the administrative contact fax number.
- **Str** accFaxNumber - The fax number of the administrative contact.
- **Str** accFaxExtention - The extension of the administrative contact fax number.
- **Int** CreationDate - The account creation time (in the Unix time format).
- **Int** TaxStatus - The type of account:
  - 1- Personal;
  - 2 - Company.
- **Int** AStatus - The current status of the account. This parameter can have the following values:
  - 0 - Active
  - 2 - Cancelled
  - 3 - Verification required
  - 4 - Not initialized
  - 10 - Credit Hold
  - 11 - Administrative Hold
  - 12 - Credit and Administrative Hold
- **Int** FullyRegistred - The parameter indicates whether a customer completed a full registration. It can have the following values:
  - 0 - No
  - 1 - Yes

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request. -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000017</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
                    <!-- Requested Account ID -->
            <value><i4>1000017</i4></value>
                    <!-- VendorAccountID -->
            <value><i4>1</i4></value>
                    <!-- CompanyName -->
            <value><string>JV SkyWings</string></value>
                    <!-- Address1 -->
            <value><string>Sunrise Valley Drive</string></value>
                    <!-- Address2 -->
            <value><string>Suite 600</string></value>
                    <!-- City -->
            <value><string>Herndon</string></value>
                    <!-- State -->
            <value><string>VA</string></value>
                    <!-- Zip -->
            <value><string>20171</string></value>
                    <!-- CountryID -->
            <value><string>us</string></value>
                    <!-- PostalAddress -->
            <value><string>
             13755 Sunrise Valley Drive, Suite 600, Herndon, VA 20171, USA
            </string></value>
                    <!-- AdminFName -->
            <value><string>John</string></value>
                    <!-- AdminMName -->
            <value><string>M.</string></value>
                    <!-- AdminLName -->
            <value><string>Smith</string></value>
                    <!-- AdminEmail -->
            <value><string>jsmith@skywings.com</string></value>
                    <!-- AdminPhCountryCode -->
            <value><string>1</string></value>
                    <!-- AdminPhAreaCode -->
            <value><string>703</string></value>
                    <!-- AdminPhNumber -->
            <value><string>8155670</string></value>
                    <!-- AdminPhExtention -->
            <value><string>111</string></value>
                    <!-- AdminFaxCountryCode -->
            <value><string>1</string></value>
                    <!-- AdminFaxAreaCode -->
            <value><string>703</string></value>
                    <!-- AdminFaxNumber -->
            <value><string>8155670</string></value>
                    <!-- AdminFaxExtention -->
            <value><string>112</string></value>
                    <!-- Account Creation Date (Unix format) -->
            <value><i4>1245244295</i4></value>
                    <!-- Account Tax Status -->
            <value><i4>2</i4></value>
                    <!-- Account Status -->
            <value><i4>0</i4></value>
                    <!-- FullyRegistered -->
            <value><i4>1</i4></value>
           </data>
          </array>
         </value>
        </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountDetailsGetEx_API

Description	Parameters	
	Passed	Returned
This method returns the details on the specified account. Which contact information is returned (a company (or person) name, email, phone number, and fax) depends on the account type: • If the account type is **Company**, the administrative contact information is returned. • If the account type is **Personal**, the personal contact information is returned.	1	27

The syntax below is given for company accounts.

Syntax

```
ItemResult BM::AccountDetailsGetEx_API(
Int AccountID.
)
returns
Int AccountID;
Int VendorAccountID;
Str CompanyName;
Str Address1;
Str Address2;
Str City;
Str State;
Str Zip;
Str CountryID;
Str PostalAddress;
Str accFName;
Str accMName;
Str accLName;
Str accEmail;
Str accPhCountryCode;
Str accPhAreaCode;
Str accPhNumber;
Str accPhExtention;
Str accFaxCountryCode;
Str accFaxAreaCode;
Str accFaxNumber;
Str accFaxExtention;
Int CreationDate;
Int TaxStatus;
Int AStatus;
Str  TaxRegID;
Int FullyRegistred.
```

Parameters description

Input parameter:

- **Int** AccountID - ID of the account for which you want to get the details.

Output parameters:

- **Int** AccountID - The account identifier.
- **Int** VendorAccountID - The identifier of the vendor (provider or reseller) account to which the specified customer account belongs.
- **Str** CompanyName - The company name (for an account of the **Company** type) or the first and last names (for a **Personal** account).
- **Str** Address1 - The account owner's address 1 (line 1/2).
- **Str** Address2 - The account owner's address 2 (line 2/2).
- **Str** City - The account owner's home city.
- **Str** State - The account owner's home state or province.
- **Str** Zip - The account owner's postal code.
- **Str** CountryID - The two-letter code of the account owner's home country.
- **Str** PostalAddress - The account owner's postal address written in the free form.
- **Str** accFName - The first name of the account owner's administrative contact.
- **Str** accMName - The middle name of the account owner's administrative contact.
- **Str** accLName - The last name of the account owner's administrative contact.
- **Str** accEmail - The email of the account owner's administrative contact.
- **Str**accPhCountryCode - The country code of the administrative contact phone number.
- **Str** accPhAreaCode - The area code of the administrative contact phone number.
- **Str** accPhNumber - The administrative contact phone number.
- **Str** accPhExtention - The extension of the administrative contact phone number.
- **Str** accFaxCountryCode - The country code of the administrative contact fax number.
- **Str** accFaxAreaCode - The area code of the administrative contact fax number.
- **Str** accFaxNumber - The fax number of the administrative contact.
- **Str** accFaxExtention - The extension of the administrative contact fax number.
- **Int** CreationDate - The account creation time (in the Unix time format).
- **Int** TaxStatus - The type of account:
  - 1- Personal;
  - 2 - Company.
- **Int** AStatus - The current status of the account. This parameter can have the following values:
  - 0 - Active
  - 2 - Cancelled
  - 3 - Verification required

    - 4 - Not initialized
    - 10 - Credit Hold
    - 11 - Administrative Hold
    - 12 - Credit and Administrative Hold
- **Str** TaxRegID - The taxpayer identifier used by the account owner for tax registration.
- **Int** FullyRegistred - The parameter indicates whether a customer completed a full registration. It can have the following values:
    - 0 - No
    - 1 - Yes

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request. -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountDetailsGetEx_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000017</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
                    <!-- Requested Account ID -->
            <value><i4>1000017</i4></value>
                    <!-- VendorAccountID -->
            <value><i4>1</i4></value>
                    <!-- CompanyName -->
            <value><string>JV SkyWings</string></value>
                    <!-- Address1 -->
            <value><string>Sunrise Valley Drive</string></value>
                    <!-- Address2 -->
            <value><string>Suite 600</string></value>
                    <!-- City -->
            <value><string>Herndon</string></value>

                    <!-- State -->
            <value><string>VA</string></value>

                    <!-- Zip -->
            <value><string>20171</string></value>

                    <!-- CountryID -->
            <value><string>us</string></value>

                    <!-- PostalAddress -->
            <value><string>
             13755 Sunrise Valley Drive, Suite 600, Herndon, VA 20171, USA
            </string></value>

                    <!-- AdminFName -->
            <value><string>John</string></value>

                    <!-- AdminMName -->
            <value><string>M.</string></value>

                    <!-- AdminLName -->
            <value><string>Smith</string></value>

                    <!-- AdminEmail -->
            <value><string>jsmith@skywings.com</string></value>

                    <!-- AdminPhCountryCode -->
            <value><string>1</string></value>

                    <!-- AdminPhAreaCode -->
            <value><string>703</string></value>

                    <!-- AdminPhNumber -->
            <value><string>8155670</string></value>

                    <!-- AdminPhExtention -->
            <value><string>111</string></value>

                    <!-- AdminFaxCountryCode -->
            <value><string>1</string></value>

                    <!-- AdminFaxAreaCode -->
            <value><string>703</string></value>

                    <!-- AdminFaxNumber -->
            <value><string>8155670</string></value>

                    <!-- AdminFaxExtention -->
            <value><string>112</string></value>

                    <!-- Account Creation Date (Unix format) -->
            <value><i4>1245244295</i4></value>

                    <!-- Account Tax Status -->
            <value><i4>2</i4></value>

                    <!-- Account Status -->
            <value><i4>0</i4></value>

                    <!-- TaxRegID -->
            <value><string>0000</string></value>

                    <!-- FullyRegistered -->
            <value><i4>1</i4></value>
           </data>
          </array>
         </value>
        </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountExtendedDetailsGet_API

	Parameters	
Description	Passed	Returned
Method returns extended details of specified account. Contact information returned (name, e-mail, phone, and fax) depends on account type.	1	47

Syntax

```
ItemResult BM :: AccountExtendedDetailsGet_API (
Int AccountID
)
returns
Int AccountID  - account ID;
Str CompanyName - company name;
Str Address1 - account addrelss 1 (line 1/2);
Str Address2 - account address 2 (line 2/2);
Str City - account city;
Str State - account state / province;
Str Zip - account postal code;
Str CountryID - account two-letter country code;
Str PostalAddress - account postal address in free form.
Str AdminFName - first name (administrative contact);
Str AdminMName - middle name (administrative contact);
Str AdminLName - last name (administrative contact);
Str AdminEmail - e-mail (administrative contact);
Str AdminPhCountryCode - phone country code (administrative contact);
Str AdminPhAreaCode - phone area code (administrative contact);
Str AdminPhNumber - phone number (administrative contact);
Str AdminPhExtention - phone extension (administrative contact);
Str AdminFaxCountryCode - fax country code (administrative contact);
Str AdminFaxAreaCode - fax area code (administrative contact);
Str AdminFaxNumber - fax number (administrative contact);
Str AdminFaxExtention - fax extension (administrative contact);
Str BillFName - first name (billing contact);
Str BillMName - middle name (billing contact);
Str BillLName - last name (billing contact);
Str BillEmail - e-mail (billing contact);
Str BillPhCountryCode - phone country code (billing contact);
Str BillPhAreaCode - phone area code (billing contact);
Str BillPhNumber - phone number (billing contact);
Str BillPhExtention - phone extension (billing contact);
Str BillFaxCountryCode - fax country code (billing contact);
Str BillFaxAreaCode - fax area code (billing contact);
Str BillFaxNumber - fax number (billing contact);
Str BillFaxExtention - fax extension (billing contact);
Str TechFName - first name (technical contact);
Str TechMName - middle name (technical contact);
Str TechLName - last name (technical contact);
Str TechEmail - e-mail (technical contact);
Str TechPhCountryCode - phone country code (technical contact);
Str TechPhAreaCode - phone area code (technical contact);
Str TechPhNumber - phone number (technical contact);
Str TechPhExtention - phone extension (technical contact);
Str TechFaxCountryCode - fax country code (technical contact);
Str TechFaxAreaCode - fax area code (technical contact);
Str TechFaxNumber - fax number (technical contact);
Str TechFaxExtention - fax extension (technical contact);
Int ClassID - customer class ID;
Str TaxZoneID - ID of tax zone to regulate account taxation.
```

Special Notes

Int AccountID - ID of account to get details for.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>AccountExtendedDetailsGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                                <!-- AccountID -->
                  <value><i4>1000015</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
```

```
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
                   <array>
                <data>
                  <value>
                    <array>
                      <data>
                                 <!-- AccountID -->
                        <value><i4>1000015</i4></value>
                                 <!-- CompanyName -->
                        <value><string>Conrad Tribble</string></value>
                                 <!-- Address 1 -->
                        <value><string>Blumen Strasse 5</string></value>
                                 <!-- Address 2 -->
                        <value><string>Apt. 17</string></value>
                                 <!-- City -->
                        <value><string>Munich</string></value>
                                 <!-- State -->
                        <value><string></string></value>
                                 <!-- Zip -->
                        <value><string>80539</string></value>
                                 <!-- CountryID -->
                        <value><string>de</string></value>
                                 <!-- PostalAddress -->
                        <value><string>Direct address</string></value>
                          <!-- Administrative Contact Section -->
                                 <!-- AdminFName -->
                        <value><string>Conrad</string></value>
                                 <!-- AdminMName -->
                        <value><string>W</string></value>
                                 <!-- AdminLName -->
                        <value><string>Tribble</string></value>
                                 <!-- AdminEmail -->
                        <value><string>email@nowhere.com</string></value>
                                 <!-- AdminPhCountryCode -->
                        <value><string>89</string></value>
                                 <!-- AdminPhAreaCode -->
                        <value><string>2888</string></value>
                                 <!-- AdminPhNumber -->
                        <value><string>112313</string></value>
                                 <!-- AdminPhExtention -->
                        <value><string></string></value>
                                 <!-- AdminFaxCountryCode -->
                        <value><string>23</string></value>
                                 <!-- AdminFaxAreaCode -->
                        <value><string>1123</string></value>
                                 <!-- AdminFaxNumber -->
                        <value><string>123</string></value>
                                 <!-- AdminFaxExtention -->
                        <value><string>d</string></value>
                          <!-- Billing Contact Section -->
                          <!-- BillFname -->
                        <value><string>Paul</string></value>
                                 <!-- BillMName -->
                        <value><string>Bill</string></value>
```

```
                                             <!-- BillLName -->
                                    <value><string>Jones</string></value>
                                             <!-- BillEmail -->
                                    <value><string>bill@nowhere.com</string></value>
                                             <!-- BillPhCountryCode -->
                                          <value><string>3</string></value>
                                             <!-- BillPhAreaCode -->
                                    <value><string>333</string></value>
                                             <!-- BillPhNumber -->
                                    <value><string>8155670</string></value>
                                             <!-- BillPhExtention -->
                                    <value><string>1</string></value>
                                             <!-- BillFaxCountryCode -->
                                    <value><string>1</string></value>
                                             <!-- BillFaxAreaCode -->
                                    <value><string>132</string></value>
                                             <!-- BillFaxNumber -->
                                    <value><string>123132</string></value>
                                             <!-- BillFaxExtention -->
                                    <value><string>2</string></value>
                                       <!-- Technical Account Section -->
                                             <!-- TechFname -->
                                    <value><string>Paul</string></value>
                                             <!-- TechMName -->
                                    <value><string>Tech</string></value>
                                             <!-- TechLName -->
                                    <value><string>Jones</string></value>
                                             <!-- TechEmail -->
                                    <value><string>techr@nowhere.com</string></value>
                                             <!-- TechPhCountryCode -->
                                    <value><string>1</string></value>
                                             <!-- TechPhAreaCode -->
                                    <value><string>23</string></value>
                                       <!-- TechPhNumber -->
                                    <value><string>2223</string></value>
                                       <!-- TechPhExtention -->
                                    <value><string>1</string></value>
                                             <!-- TechFaxCountryCode -->
                                    <value><string>1</string></value>
                                             <!-- TechFaxAreaCode -->
                                    <value><string>123</string></value>
                                             <!-- TechFaxNumber -->
                                    <value><string>123</string></value>
                                             <!-- TechFaxExtention -->
                                    <value><string>232</string></value>
                                             <!-- ClassID -->
                                    <value><i4>0</i4></value>
                                             <!-- TaxZoneID -->
                                          <value>Default</value>
                                  </data>
                                </array>
                              </value>
                            </data>
                          </array>
                        </value>
                      </member>
                    </struct>
                  </value>
                </param>
```

```
  </params>
</methodResponse>
```

AccountPutOnAdmHold_API

Description	Parameters	
	Passed	**Returned**
The method puts an account on *Administrative hold.* The account subscriptions get the status "Administrative Hold", service status is "Stopped", and the account status if "Administrative Hold".	1	1

Syntax

```
ItemResult BM::AccountPutOnAdmHold_API(
  Int AccountID.
)
returns
  Str ErrorMessage.
```

Parameters Description

Input Parameter:

Int AccountID – Identifier of an account which is to be put on hold.

Output Parameter:

Str ErrorMessage – Message issued after the specified account is put on the *Administrative hold* successfully. The message content is as follows: "Account is put on administrative hold."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountPutOnAdmHold_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Account was put on administrative hold.
              </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AccountTakeFromAdmHold_API

	Parameters	
Description	Passed	Returned

The method releases an account from *Administrative hold.* The account subscriptions get the status "Active", service status is "Running", and the account status is "Active".	1	1

Syntax

```
ItemResult BM::AccountTakeFromAdmHold_API(
  Int AccountID.
)
returns
  Str ErrorMessage.
```

Parameters Description

Input Parameter:

Int AccountID – Identifier of an account that is to be released from the *Administrative hold*.

Output Parameter:

Str ErrorMessage – Message that is displayed when the specified account is released from the *Administrative hold* successfully. The message content is the following: "Account was released from hold."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountTakeFromAdmHold_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
        <!-- AccountID -->
         <value><i4>123</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Account was released from hold.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

AccountUpdate_API

Description	Parameters	
	Passed	Returned
This method updates information for the specified account. There are two types of account in PBA: personal and company. Company accounts need a company name, address and three types of contacts (administrative, billing and technical) to be passed. Personal accounts need only an address and personal details. Personal details are represented by the administrative contact in this method. The method request example is given for the company account type. If you do not want to change values of some parameters, pass the old values for those parameters. **Note**: The method does not propagate information to POA (information on updating the account information will not be transferred to POA).	46	1

Syntax

```
ErrorMessage BM::AccountUpdate_API(
  Int AccountID;
  Str(80) CompanyName;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(1024) PostalAddress;
  Str(30) AdminFName;
  Str(30) AdminMName;
  Str(30) AdminLName;
  Str(100) AdminEmail;
  Str(4) AdminPhCountryCode;
  Str(10) AdminPhAreaCode;
  Str(20) AdminPhNumber;
  Str(10) AdminPhExtention;
  Str(4) AdminFaxCountryCode;
  Str(10) AdminFaxAreaCode;
  Str(20) AdminFaxNumber;
  Str(10) AdminFaxExtention;
  Str(30) BillFName;
  Str(30) BillMName;
  Str(30) BillLName;
  Str(100) BillEmail;
  Str(4) BillPhCountryCode;
  Str(10) BillPhAreaCode;
  Str(20) BillPhNumber;
  Str(10) BillPhExtention;
  Str(4) BillFaxCountryCode;
  Str(10) BillFaxAreaCode;
  Str(20) BillFaxNumber;
  Str(10) BillFaxExtention;
  Str(30) TechFName;
  Str(30) TechMName;
  Str(30) TechLName;
  Str(100) TechEmail;
  Str(4) TechPhCountryCode;
  Str(10) TechPhAreaCode;
  Str(20) TechPhNumber;
  Str(10) TechPhExtention;
  Str(4) TechFaxCountryCode;
  Str(10) TechFaxAreaCode;
  Str(20) TechFaxNumber;
  Str(10) TechFaxExtention;
  Int ClassID.
)
returns
  Str ErrorMessage.
```

Parameters description

Note: If you are updating a personal account information, the company-specific parameters must be passed as empty (zero) values, and vice versa.

Input parameters:

The following parameters are relevant for both types of account (company and person).

- **Int** AccountID - The identifier of the account that is to be updated.
- **Str(80)** CompanyName - The account name. It can be a company name or a person's first and last names.
- **Str(80)** Address1 - The account owner's address (line 1/2).
- **Str(80)** Address2 - The account owner's address (line 2/2).
- **Str(40)** City - The account owner's home city.
- **Str(80)** State - The account owner's home state.
- **Str(10)** Zip - The account owner's zip or postal code. It will be verified through a country specific regular expression. You can modify the regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the PBA control panel.
- **Str(2)** CountryID - The two-letter code of the account owner's home country. This code is verified against the countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the PBA control panel.
- **Str(1024)** PostalAddress - The account owner's postal address written in the free form.

The following parameters are relevant to both types of account (company and personal). If an account of the company type is updated, these parameters are used as the account administrative contact information. If an account of the person type is updated, these parameters are used as the personal contact information.

- **Str(30)** AdminFName - The first name of the account owner's administrative contact.
- **Str(30)** AdminMName - The middle name of the account owner's administrative contact.
- **Str(30)** AdminLName - The last name of the account owner's administrative contact.
- **Str(100)** AdminEmail - The email of the account owner's administrative contact.
- **Str(4)** AdminPhCountryCode - The country code of the administrative contact phone number.
- **Str(10)** AdminPhAreaCode - The area code of the administrative contact phone number.
- **Str(20)** AdminPhNumber - The administrative contact phone number.
- **Str(10)** AdminPhExtention - The extension of the administrative contact phone number.
- **Str(4)** AdminFaxCountryCode - The country code of the administrative contact fax number.
- **Str(10)** AdminFaxAreaCode - The area code of the administrative contact fax number.
- **Str(20)** AdminFaxNumber - The fax number of the administrative contact.
- **Str(10)** AdminFaxExtention - The extension of the administrative contact fax number.

The following parameters are relevant to accounts of the company type. For the personal account type they must be passed empty.

- **Str(30)** BillFName - The first name of the account owner's billing contact.
- **Str(30)** BillMName - The middle name of the account owner's billing contact.
- **Str(30)** BillLName - The last name of the account owner's billing contact.
- **Str(100)** BillEmail - The email address of the account owner's billing contact.
- **Str(4)** BillPhCountryCode - The country code of the billing contact phone number.
- **Str(10)** BillPhAreaCode - The area code of the billing contact phone number.
- **Str(20)** BillPhNumber - The billing contact phone number.
- **Str(10)** BillPhExtention - The extension of the billing contact phone number.
- **Str(4)** BillFaxCountryCode - The country code of the billing contact fax number.
- **Str(10)** BillFaxAreaCode - The area code of the billing contact fax number.
- **Str(20)** BillFaxNumber - The fax number of the billing contact.
- **Str(10)** BillFaxExtention - The extension of the billing contact fax number.
- **Str(30)** TechFName - The first name of the account owner's technical contact.
- **Str(30)** TechMName - The middle name of the account owner's technical contact.

- **Str(30)** TechLName - The last name of the account owner's technical contact.
- **Str(100)** TechEmail - The email address of the account owner's technical contact.
- **Str(4)** TechPhCountryCode - The country code of the technical contact phone number.
- **Str(10)** TechPhAreaCode - The area code of the technical contact phone number.
- **Str(20)** TechPhNumber - The technical contact phone number.
- **Str(10)** TechPhExtention - The extension of the technical contact phone number.
- **Str(4)** TechFaxCountryCode - The country code of the technical contact fax number.
- **Str(10)** TechFaxAreaCode - The area code of the technical contact fax number.
- **Str(20)** TechFaxNumber - The fax number of the technical contact.
- **Str(10)** TechFaxExtention - The extension of the technical contact fax number.

The following parameter is relevant to both types of account (company and personal).

- **Int** ClassID - The identifier of the class to which the customer belongs. The account current credit terms will be updated with the credit terms assigned to the customer class specified. You can find the list of available customer classes in the **System** > **Settings** > **Customer Classes** submenu of the PBA control panel.

Output parameter:

- **Str** ErrorMessage - The message displayed when the account information is updated: "Operation done."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request. -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000001</i4></value>
         <!-- Company -->
         <value>JV SkyWings</value>
         <!-- Address1 -->
         <value>Sunrise Valley Drive</value>
         <!-- Address2 -->
         <value>Suite 600</value>
         <!-- City -->
         <value>Herndon</value>
         <!-- State -->
         <value>VA</value>
         <!-- Zip -->
         <value>20171</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PostalAddress -->
         <value>
          13755 Sunrise Valley Drive, Suite 600, Herndon, VA 20171, USA
         </value>
         <!-- AdminFName -->
         <value>John</value>
         <!-- AdminMName -->
         <value>M.</value>
         <!-- AdminLName -->
         <value>Smith</value>
         <!-- AdminEmail -->
         <value>johnsmith@skywings.com</value>
         <!-- AdminPhCountryCode -->
         <value>1</value>
         <!-- AdminPhAreaCode -->
         <value>703</value>
```

```
<!-- AdminPhNumber -->
<value>8155670</value>
<!-- AdminPhExtention -->
<value>111</value>
<!-- AdminFaxCountryCode -->
<value>1</value>
<!-- AdminFaxAreaCode -->
<value>703</value>
<!-- AdminFaxNumber -->
<value>8155670</value>
<!-- AdminFaxExtention -->
<value>111</value>
<!-- BillFName -->
<value>John</value>
<!-- BillMName -->
<value>M.</value>
<!-- BillLName -->
<value>Smith</value>
<!-- BillEmail -->
<value>johnsmith@skywings.com</value>
<!-- BillPhCountryCode -->
<value>1</value>
<!-- BillPhAreaCode -->
<value>703</value>
<!-- BillPhNumber -->
<value>8155670</value>
<!-- BillPhExtention -->
<value>111</value>
<!-- BillFaxCountryCode -->
<value>1</value>
<!-- BillFaxAreaCode -->
<value>703</value>
<!-- BillFaxNumber -->
<value>8155670</value>
<!-- BillFaxExtention -->
<value>111</value>
<!-- TechFName -->
<value>John</value>
<!-- TechMName -->
<value>M.</value>
<!-- TechLName -->
<value>Smith</value>
<!-- TechEmail -->
<value>johnsmith@skywings.com</value>
<!-- TechPhCountryCode -->
<value>1</value>
<!-- TechPhAreaCode -->
<value>703</value>
<!-- TechPhNumber -->
<value>8155670</value>
<!-- TechPhExtention -->
<value>111</value>
<!-- TechFaxCountryCode -->
<value>1</value>
<!-- TechFaxAreaCode -->
<value>703</value>
<!-- TechFaxNumber -->
<value>8155670</value>
```

```
         <!-- TechFaxExtention -->
         <value>111</value>
         <!-- ClassID -->
         <value><i4>12</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

AccountUpdateEx_API

Description	Parameters	
	Passed	Returned
This method updates information for the specified account. There are two types of account in PBA: personal and company. Company accounts need a company name, address and three types of contacts (administrative, billing and technical) to be passed. Personal accounts need only an address and personal details. Personal details are represented by the administrative contact in this method. The method request example is given for the company account type. If you do not want to change values of some parameters, pass the old values for those parameters. **Note**: The method does not propagate information to POA (information on updating the account information will not be transferred to POA).	47	1

Syntax

```
ErrorMessage BM::AccountUpdateEx_API(
  Int AccountID;
  Str(80) CompanyName;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(1024) PostalAddress;
  Str(30) AdminFName;
  Str(30) AdminMName;
  Str(30) AdminLName;
  Str(100) AdminEmail;
  Str(4) AdminPhCountryCode;
  Str(10) AdminPhAreaCode;
  Str(20) AdminPhNumber;
  Str(10) AdminPhExtention;
  Str(4) AdminFaxCountryCode;
  Str(10) AdminFaxAreaCode;
  Str(20) AdminFaxNumber;
  Str(10) AdminFaxExtention;
  Str(30) BillFName;
  Str(30) BillMName;
  Str(30) BillLName;
  Str(100) BillEmail;
  Str(4) BillPhCountryCode;
  Str(10) BillPhAreaCode;
  Str(20) BillPhNumber;
  Str(10) BillPhExtention;
  Str(4) BillFaxCountryCode;
  Str(10) BillFaxAreaCode;
  Str(20) BillFaxNumber;
  Str(10) BillFaxExtention;
  Str(30) TechFName;
  Str(30) TechMName;
  Str(30) TechLName;
  Str(100) TechEmail;
  Str(4) TechPhCountryCode;
  Str(10) TechPhAreaCode;
  Str(20) TechPhNumber;
  Str(10) TechPhExtention;
  Str(4) TechFaxCountryCode;
  Str(10) TechFaxAreaCode;
  Str(20) TechFaxNumber;
  Str(10) TechFaxExtention;
  Int ClassID;
  Str TaxRegID.
)
returns
  Str ErrorMessage
```

Parameters description

Note: If you are updating a personal account information, the company-specific parameters must be passed as empty (zero) values, and vice versa.

Input parameters:

The following parameters are relevant for both types of account (company and person).

- **Int** AccountID - The identifier of the account that is to be updated.
- **Str(80)** CompanyName - The account name. It can be a company name or a person's first and last names.
- **Str(80)** Address1 - The account owner's address (line 1/2).
- **Str(80)** Address2 - The account owner's address (line 2/2).
- **Str(40)** City - The account owner's home city.
- **Str(80)** State - The account owner's home state.
- **Str(10)** Zip - The account owner's zip or postal code. It will be verified through a country specific regular expression. You can modify the regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the PBA control panel.
- **Str(2)** CountryID - The two-letter code of the account owner's home country. This code is verified against the countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the PBA control panel.

- **Str(1024)** PostalAddress - The account owner's postal address written in the free form.

The following parameters are relevant to both types of account (company and personal). If an account of the company type is updated, these parameters are used as the account administrative contact information. If an account of the person type is updated, these parameters are used as the personal contact information.

- **Str(30)** AdminFName - The first name of the account owner's administrative contact.
- **Str(30)** AdminMName - The middle name of the account owner's administrative contact.
- **Str(30)** AdminLName - The last name of the account owner's administrative contact.
- **Str(100)** AdminEmail - The email of the account owner's administrative contact.
- **Str(4)** AdminPhCountryCode - The country code of the administrative contact phone number.
- **Str(10)** AdminPhAreaCode - The area code of the administrative contact phone number.
- **Str(20)** AdminPhNumber - The administrative contact phone number.
- **Str(10)** AdminPhExtention - The extension of the administrative contact phone number.
- **Str(4)** AdminFaxCountryCode - The country code of the administrative contact fax number.
- **Str(10)** AdminFaxAreaCode - The area code of the administrative contact fax number.
- **Str(20)** AdminFaxNumber - The fax number of the administrative contact.
- **Str(10)** AdminFaxExtention - The extension of the administrative contact fax number.

The following parameters are relevant to accounts of the company type. For the personal account type they must be passed empty.

- **Str(30)** BillFName - The first name of the account owner's billing contact.
- **Str(30)** BillMName - The middle name of the account owner's billing contact.
- **Str(30)** BillLName - The last name of the account owner's billing contact.
- **Str(100)** BillEmail - The email address of the account owner's billing contact.
- **Str(4)** BillPhCountryCode - The country code of the billing contact phone number.
- **Str(10)** BillPhAreaCode - The area code of the billing contact phone number.
- **Str(20)** BillPhNumber - The billing contact phone number.
- **Str(10)** BillPhExtention - The extension of the billing contact phone number.
- **Str(4)** BillFaxCountryCode - The country code of the billing contact fax number.
- **Str(10)** BillFaxAreaCode - The area code of the billing contact fax number.
- **Str(20)** BillFaxNumber - The fax number of the billing contact.
- **Str(10)** BillFaxExtention - The extension of the billing contact fax number.
- **Str(30)** TechFName - The first name of the account owner's technical contact.

- **Str(30)** TechMName - The middle name of the account owner's technical contact.
- **Str(30)** TechLName - The last name of the account owner's technical contact.
- **Str(100)** TechEmail - The email address of the account owner's technical contact.
- **Str(4)** TechPhCountryCode - The country code of the technical contact phone number.
- **Str(10)** TechPhAreaCode - The area code of the technical contact phone number.
- **Str(20)** TechPhNumber - The technical contact phone number.
- **Str(10)** TechPhExtention - The extension of the technical contact phone number.
- **Str(4)** TechFaxCountryCode - The country code of the technical contact fax number.
- **Str(10)** TechFaxAreaCode - The area code of the technical contact fax number.
- **Str(20)** TechFaxNumber - The fax number of the technical contact.
- **Str(10)** TechFaxExtention - The extension of the technical contact fax number..

The following parameters are relevant to both types of account (company and personal).

- **Int** ClassID - The identifier of the class to which the customer belongs. The account current credit terms will be updated with the credit terms assigned to the customer class specified. You can find the list of available customer classes in the **System** > **Settings** > **Customer Classes** submenu of the PBA control panel.
- **Str** TaxRegID - The taxpayer identifier used by the account owner for tax registration.

Output parameter:

- **Str** ErrorMessage - The message displayed when the account information is updated: "Operation done."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request. -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountUpdateEx_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000001</i4></value>
         <!-- Company -->
         <value>JV SkyWings</value>
         <!-- Address1 -->
         <value>Sunrise Valley Drive</value>
         <!-- Address2 -->
         <value>Suite 600</value>
         <!-- City -->
         <value>Herndon</value>
         <!-- State -->
         <value>VA</value>
         <!-- Zip -->
         <value>20171</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PostalAddress -->
         <value>
          13755 Sunrise Valley Drive, Suite 600, Herndon, VA 20171, USA
         </value>
         <!-- AdminFName -->
         <value>John</value>
         <!-- AdminMName -->
         <value>M.</value>
         <!-- AdminLName -->
         <value>Smith</value>
         <!-- AdminEmail -->
         <value>johnsmith@skywings.com</value>
         <!-- AdminPhCountryCode -->
         <value>1</value>
         <!-- AdminPhAreaCode -->
         <value>703</value>
```

```
<!-- AdminPhNumber -->
<value>8155670</value>
<!-- AdminPhExtention -->
<value>111</value>
<!-- AdminFaxCountryCode -->
<value>1</value>
<!-- AdminFaxAreaCode -->
<value>703</value>
<!-- AdminFaxNumber -->
<value>8155670</value>
<!-- AdminFaxExtention -->
<value>111</value>
<!-- BillFName -->
<value>John</value>
<!-- BillMName -->
<value>M.</value>
<!-- BillLName -->
<value>Smith</value>
<!-- BillEmail -->
<value>johnsmith@skywings.com</value>
<!-- BillPhCountryCode -->
<value>1</value>
<!-- BillPhAreaCode -->
<value>703</value>
<!-- BillPhNumber -->
<value>8155670</value>
<!-- BillPhExtention -->
<value>111</value>
<!-- BillFaxCountryCode -->
<value>1</value>
<!-- BillFaxAreaCode -->
<value>703</value>
<!-- BillFaxNumber -->
<value>8155670</value>
<!-- BillFaxExtention -->
<value>111</value>
<!-- TechFName -->
<value>John</value>
<!-- TechMName -->
<value>M.</value>
<!-- TechLName -->
<value>Smith</value>
<!-- TechEmail -->
<value>johnsmith@skywings.com</value>
<!-- TechPhCountryCode -->
<value>1</value>
<!-- TechPhAreaCode -->
<value>703</value>
<!-- TechPhNumber -->
<value>8155670</value>
<!-- TechPhExtention -->
<value>111</value>
<!-- TechFaxCountryCode -->
<value>1</value>
<!-- TechFaxAreaCode -->
<value>703</value>
<!-- TechFaxNumber -->
<value>8155670</value>
```

```
         <!-- TechFaxExtention -->
         <value>111</value>

         <!-- ClassID -->
         <value><i4>12</i4></value>

         <!-- TaxRegID -->
         <value><string>0000</string></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

AccountWithIDAdd_API

Description	Parameters	
	Passed	**Returned**
Starting PBA 4.4.2 the method is obsolete and left for backward compatibility, use `AccountAdd_API` method instead. Method creates new account in PBA with specified ID. There are two types of account in PBA: person and company. Company account needs company name, address and three types of contacts (administrative, billing and technical) to be passed. Personal account needs address and personal details only. Example below is given for person account type.	62	1

Syntax

```
ErrorMessage BM::AccountWithIDAdd_API(
  Int VendorAccountID;
  Int VType;
  Str(80) CompanyName;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(1024) PostalAddress;
  Str(30) AdminFName;
  Str(30) AdminMName;
  Str(30) AdminLName;
  Str(100) AdminEmail;
  Str(4) AdminPhCountryCode;
  Str(10) AdminPhAreaCode;
  Str(20) AdminPhNumber;
  Str(10) AdminPhExtention;
  Str(4) AdminFaxCountryCode;
  Str(10) AdminFaxAreaCode;
  Str(20) AdminFaxNumber;
  Str(10) AdminFaxExtention;
  Str(30) BillFName;
  Str(30) BillMName;
  Str(30) BillLName;
  Str(100) BillEmail;
  Str(4) BillPhCountryCode;
  Str(10) BillPhAreaCode;
  Str(20) BillPhNumber;
  Str(10) BillPhExtention;
  Str(4) BillFaxCountryCode;
  Str(10) BillFaxAreaCode;
  Str(20) BillFaxNumber;
  Str(10) BillFaxExtention;
  Str(30) TechFName;
  Str(30) TechMName;
  Str(30) TechLName;
  Str(100) TechEmail;
  Str(4) TechPhCountryCode;
  Str(10) TechPhAreaCode;
  Str(20) TechPhNumber;
  Str(10) TechPhExtention;
  Str(4) TechFaxCountryCode;
  Str(10) TechFaxAreaCode;
  Str(20) TechFaxNumber;
  Str(10) TechFaxExtention;
  Str(30) PersFName;
  Str(30) PersMName;
  Str(30) PersLName;
  Int Birthday;
  Str(1024) Passport;
  Str(100) PersEmail;
  Str(4) PersPhCountryCode;
  Str(10) PersPhAreaCode;
  Str(20) PersPhNumber;
  Str(10) PersPhExtention;
  Str(4) PersFaxCountryCode;
  Str(10) PersFaxAreaCode;
  Str(20) PersFaxNumber;
  Str(10) PersFaxExtention;
  Int TaxStatus;
  Int AccountID.
)
returns
  Str ErrorMessage - message after account has been added: " Account # is
successfully added."
```

Special Notes

Note: Company specific parameters should be passed as empty/zero, if personal account is added and vice versa.

The following parameters are relevant for both types of account (company and person).

- **Int** VendorAccountID - vendor account ID (provider or reseller);
- **Int** VType - account role: 2 - Reseller, 3 - Customer;
- **Str(80)** CompanyName - account name. It can be company name or person first and last names.
- **Str(80)** Address1 - account address (line 1/2);
- **Str(80)** Address2 - account address (line 2/2);
- **Str(40)** City - account address: city;
- **Str(80)** State - account address: state;

- **Str(10)** Zip - account address: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **Configuration Director** > **Miscellaneous Settings** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - two-letter country code, verified against countries list in **Configuration Director** > **Miscellaneous Settings** > **Countries** submenu of the Navigation tree;
- **Str(1024)** PostalAddress - account address in free form;

The following parameters are relevant for company account type. For person account type should be passed empty.

- **Str(30)** AdminFName - account administrative contact information: first name;
- **Str(30)** AdminMName - account administrative contact information: middle name;
- **Str(30)** AdminLName - account administrative contact information: last name;
- **Str(100)** AdminEmail - account administrative contact information: e-mail address;
- **Str(4)** AdminPhCountryCode - account administrative contact information: phone country code;
- **Str(10)** AdminPhAreaCode - account administrative contact information: phone area code;
- **Str(20)** AdminPhNumber - account administrative contact information: phone number;
- **Str(10)** AdminPhExtention - account administrative contact information: phone extension;
- **Str(4)** AdminFaxCountryCode - account administrative contact information: fax country code;
- **Str(10)** AdminFaxAreaCode - account administrative contact information: fax area code;
- **Str(20)** AdminFaxNumber - account administrative contact information: fax number;
- **Str(10)** AdminFaxExtention - account administrative contact information: fax extension;
- **Str(30)** BillFName - account billing contact information: first name;
- **Str(30)** BillMName - account billing contact information: middle name;
- **Str(30)** BillLName - account billing contact information: last name;
- **Str(100)** BillEmail - account billing contact information: e-mail address;
- **Str(4)** BillPhCountryCode - account billing contact information: phone country code;
- **Str(10)** BillPhAreaCode - account billing contact information: phone area code;
- **Str(20)** BillPhNumber - account billing contact information: phone number;
- **Str(10)** BillPhExtention - account billing contact information: phone extension;
- **Str(4)** BillFaxCountryCode - account billing contact information: fax country code;
- **Str(10)** BillFaxAreaCode - account billing contact information: fax area code;
- **Str(20)** BillFaxNumber - account billing contact information: fax number;

- **Str(10)** BillFaxExtention - account billing contact information: fax extension;
- **Str(30)** TechFName - account technical contact information: first name;
- **Str(30)** TechMName - account technical contact information: middle name;
- **Str(30)** TechLName - account technical contact information: last name;
- **Str(100)** TechEmail - account technical contact information: e-mail address;
- **Str(4)** TechPhCountryCode - account technical contact information: phone country code;
- **Str(10)** TechPhAreaCode - account technical contact information: phone area Code;
- **Str(20)** TechPhNumber - account technical contact information: phone number;
- **Str(10)** TechPhExtention - account technical contact information: phone extension;
- **Str(4)** TechFaxCountryCode - account technical contact information: fax country code;
- **Str(10)** TechFaxAreaCode - Account Tech contact information: Fax Area Code;
- **Str(20)** TechFaxNumber - account technical contact information: fax number;
- **Str(10)** TechFaxExtention - account technical contact information: fax extension;

The following parameters are relevant for person account type. For company account type should be passed empty.

- **Str(30)** PersFName - account personal contact information: first name;
- **Str(30)** PersMName - account personal contact information: middle name;
- **Str(30)** PersLName - account personal contact information: last name;
- **Int** Birthday - account personal contact information: birthday in Unix date format;
- **Str(1024)** Passport - account personal contact information: passport details;
- **Str(100)** PersEmail - account personal contact information: e-mail address;
- **Str(4)** PersPhCountryCode - account personal contact information: phone country code;
- **Str(10)** PersPhAreaCode - account personal contact information: phone area code;
- **Str(20)** PersPhNumber - account personal contact information: phone number;
- **Str(10)** PersPhExtention - account personal contact information: phone extension;
- **Str(4)** PersFaxCountryCode - account personal contact information: fax country code;
- **Str(10)** PersFaxAreaCode - account personal contact information: fax area code;
- **Str(20)** PersFaxNumber - account personal contact information: fax number;
- **Str(10)** PersFaxExtention - account personal contact information: fax extension;

The following parameters are relevant for both types of account (company and person).

- **Int** TaxStatus - account type: 1 - Person, 2 - Company;
- **Int** AccountID - account ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AccountWithIDAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- VType -->
         <value><i4>3</i4></value>
         <!-- CompanyName -->
         <value>John Smith</value>
         <!-- Address1 -->
         <value>Sunrise Valley Drive</value>
         <!-- Address2 -->
         <value>Suite 600</value>
         <!-- City -->
         <value>Herndon</value>
         <!-- State -->
         <value>VA</value>
         <!-- Zip -->
         <value>20171</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PostalAddress -->
         <value>13755 Sunrise Valley Drive, Suite 600, Herndon, VA 20171,
USA</value>
         <!-- AdminFName -->
         <value></value>
         <!-- AdminMName -->
         <value></value>
         <!-- AdminLName -->
         <value></value>
         <!-- AdminEmail -->
         <value></value>
         <!-- AdminPhCountryCode -->
         <value></value>
```

```
<!-- AdminPhAreaCode -->
<value></value>

<!-- AdminPhNumber -->
<value></value>

<!-- AdminPhExtention -->
<value></value>

<!-- AdminFaxCountryCode -->
<value></value>

<!-- AdminFaxAreaCode -->
<value></value>

<!-- AdminFaxNumber -->
<value></value>

<!-- AdminFaxExtention -->
<value></value>

<!-- BillFName -->
<value></value>

<!-- BillMName -->
<value></value>

<!-- BillLName -->
<value></value>

<!-- BillEmail -->
<value></value>

<!-- BillPhCountryCode -->
<value></value>

<!-- BillPhAreaCode -->
<value></value>

<!-- BillPhNumber -->
<value></value>

<!-- BillPhExtention -->
<value></value>

<!-- BillFaxCountryCode -->
<value></value>

<!-- BillFaxAreaCode -->
<value></value>

<!-- BillFaxNumber -->
<value></value>

<!-- BillFaxExtention -->
<value></value>

<!-- TechFName -->
<value>''John''</value>

<!-- TechMName -->
<value></value>

<!-- TechLName -->
<value></value>

<!-- TechEmail -->
<value></value>

<!-- TechPhCountryCode -->
<value></value>

<!-- TechPhAreaCode -->
<value></value>

<!-- TechPhNumber -->
<value></value>

<!-- TechPhExtention -->
<value></value>

<!-- TechFaxCountryCode -->
<value></value>

<!-- TechFaxAreaCode -->
<value></value>
```

```
          <!-- TechFaxNumber -->
          <value></value>
          <!-- TechFaxExtention -->
          <value></value>
          <!-- PersFName -->
          <value>John</value>
          <!-- PersMName -->
          <value>M.</value>
          <!-- PersLName -->
          <value>Smith</value>
          <!-- Birthday -->
          <value><i4>1161373724</i4></value>
          <!-- Passport -->
          <value>1161373724</value>
          <!-- PersEmail -->
          <value>johnsmith@yahoo.com</value>
          <!-- PersPhCountryCode -->
          <value>1</value>
         <!-- PersPhAreaCode -->
          <value>703</value>
          <!-- PersPhNumber -->
          <value>8155670</value>
          <!-- PersPhExtention -->
          <value>111</value>
          <!-- PersFaxCountryCode -->
          <value>1</value>
          <!-- PersFaxAreaCode -->
          <value>703</value>
         <!-- PersFaxNumber -->
          <value>8155670</value>
          <!-- PersFaxExtention -->
          <value>112</value>
         <!-- TaxStatus -->
          <value><i4>2</i4></value>
          <!-- AccountID -->
          <value><i4>1000001</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
```

```
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
               Account #1000001 is successfully added.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

AccountVerifyData_API

• Description	Parameters	
	Passed	**Returned**
This method verifies contact data of a newly created account. Use this method to check whether the data of the account you intend to create are valid. For the additional account attributes the method returns the additional attributes values. To verify the data (e.g., login and password) of an existing account, use the UserForVendorValidate_API (on page 642) method.	The number of passed parameters depends on the structure of data.	The list of passed additional account attributes.

Syntax

```
ItemResult BM::AccountVerifyData_API(
  Int VendorAccountID;
  Str ContactDataSlot_1;
    ...
  Str ContactDataSlot_N;
  )
returns
  an array of strings (Str).
```

Parameters Description

Input Parameters:

- **Int** VendorAccountID - account identifier of a vendor (a provider or reseller).
- **Str** ContactDataSlot - customer contact details. This information is used to fill the customer account information and information for the created default user. The contact information is submitted as an array of strings of the following format:

  <ContactDataSlotName>=<ContactDataSlotValue>

  Where:

  - <ContactDataSlotName> is the name of a contact data slot;
  - <ContactDataSlotValue> is the value of a contact data slot;

  Parallels Business Automation supports the following contact data slots:

  - LoginID - login to CP;
  - PasswordID - password to CP
  - CompanyNameID - company name for company accounts, or the first name and last name of a customer for personal accounts;

- CompanyNameLatinID - the same parameter, as 'CompanyNameID', but in the ASCII charset (for international use);
- PostAddressID - customer postal address written in a free-form;
- TaxRegID - customer tax registration identifier;
- FirstNameID - customer first name;
- MiddleNameID - customer middle name;
- LastNameID - customer last name;
- AddressID - customer address line 1;
- Address2ID - customer address line 2;
- CityID - customer residential city;
- ZipID - customer ZIP (or postal) code;
- CountryID - customer residential country;
- StateID - customer residential state or province;
- EmailID - customer e-mail address;
- PhoneCountryID - customer country phone code;
- PhoneAreaID - customer area phone code;
- PhoneNumberID - customer phone number
- PhoneExtensionID - customer phone number extension;
- FaxCountryID - customer country fax code;
- FaxAreaID - customer area fax code;
- FaxNumberID - customer fax number;
- FaxExtensionID - customer fax number extension;
- BannerID - identifier of a campaign, is applicable if the customer has been redirected to the store by one of the banner campaigns configured in PBA;
- PromoCodeID - promotion code, is applicable if the customer enters one;

Note: Each additional attribute that you create becomes a contact data slot and can be passed together with other attributes.

Output Parameters:

- An array of strings (**Str)** containing the validated values of additional account attributes.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the xml comments, REMOVE all the comments from an actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>AccountVerifyData_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <!-- VendorAccountID -->
           <value><i4>1</i4></value>
           <!-- Login to CP -->
           <value>LoginID=test123</value>
           <!-- Password to CP -->
           <value>XXXPasswordID=123test</value>
           <!-- Customer's description -->
           <value>CompanyNameID=John "tailor" Smith</value>
           <!-- First Name -->
           <value>FirstNameID=John</value>
           <!-- Middle Name -->
           <value>MiddleNameID=Shawn</value>
           <!-- Last Name -->
           <value>LastNameID=Smith</value>
           <!-- Address (line 1/2) -->
           <value>AddressID=Sunrise Valley Drive</value>
           <!-- Address (line 2/2) -->
           <value>Address2ID=Suite 600</value>
           <!-- City -->
           <value>CityID=New York</value>
           <!-- State -->
           <value>StateID=NY</value>
           <!-- Zip code -->
           <value>ZipID=12345</value>
           <!-- Country -->
           <value>CountryID=us</value>
           <!-- Email -->
           <value>EmailID=jsmith@tailor.com</value>
           <!-- Phone country code -->
           <value>PhoneCountryID=1</value>
           <!-- Phone area code -->
           <value>PhoneAreaID=201</value>

           <!-- Phone number -->
           <value>PhoneNumberID=4568523</value>

           <!-- Phone Extension -->
           <value>PhoneExtensionID=245</value>

           <!-- Fax country code -->
           <value>FaxCountryID=1</value>

           <!-- Fax area code -->
           <value>FaxAreaID=201</value>

           <!-- Fax number -->
           <value>FaxNumberID=4568523</value>

           <!-- Fax Extension -->
           <value>FaxExtensionID=235</value>

           <!-- Promo code -->
           <value>PromoCodeID=promo895</value>

           <!-- CNPJ customer tax payer's identifier (for companies). ADDITIONALLY CREATED PARAMETER.-->
           <value>CNPJ=20635245000100</value>

           <!-- CPF customer tax payer's identifier (for individuals). ADDITIONALLY CREATED PARAMETER.-->
           <value>CPF=42262487677</value>

           <!-- CLE customer identifier. ADDITIONALLY CREATED PARAMETER.-->
           <value>CLE=533352589102452</value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <value><string>CLE</string></value>
                <value><string>533352589102452</string></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><string>CNPJ</string></value>
                <value><string>20.635.245/0001-00</string></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><string>CPF</string></value>
                <value><string>422.624.876-77</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AddCreditMemo_API

Description	Parameters	
	Passed	Returned
The method adds the Credit Memo.	7	1

Syntax

```
ItemResult BM::AddCreditMemoAPI(
  Int AccountID;
  Str DocNum;
  Double Amount;
  Str Desc;
  Str TaxCatID;
  Int BranchID;
  Int SalesID.
)
returns
  Int DocumentID - ID of newly created Credit Memo (do not mix with Document
Number);
```

Special Notes

Int AccountID - Identifier of an account.

Str DocNum - Document Number (document number can be NULL as long as some Enumerator class is assigned to "Credit Memo"). This parameter is only used if no Enumerator class is assigned to Credit Memo. In other cases it is optional.

Double Amount - Credit Memo amount; should be positive.

Str Description - Description of the Credit Memo. Empty value can be passed, such as <value></value>.

Str Tax Category ID - ID of the Tax Category applied to the Credit Memo. Empty value can be passed, such as <value></value>.

Int Sales Branch ID - ID of the Sales Branch assigned to the Credit Memo. Empty value can be passed, such as <value></value>.

Int Sales Person ID - ID of the Sales Person assigned to the Credit Memo. Empty value can be passed, such as <value></value>.

Example

The request adds Credit Memo for the Account ID 1000037, Document Number not passed, for amount of 10.00, with description "Description", Tax Category ID 4, .Sales Branch ID 1, Sales Person ID 2.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>AddCreditMemo_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <value>1000037</value>
           <value></value>
           <value>10.00</value>
           <value>Description</value>
           <value>4</value>
           <value>1</value>
           <value>2</value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <!--Placed Credit Memo ID -->
             <value><i4>562</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AddPEMSubscription_API

Description	Parameters	
	Passed	**Returned**
Method creates new subscription based on given parameters. Subscription information is added to PBA database and provisioning specific information (for example, resource rates) is added to PEMGATE table.	19	1

Syntax

```
ItemResult PEMGATE::AddPEMSubscription_API(
  Int Status;
  Int ServStatus;
  Int StartDate;
  Int ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int Period;
  Int PeriodType;
  Str DomainName;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate.
)
returns
  Int SubscriptionID - created subscription ID.
```

Special Notes

- **Int** Status - parent subscription status (10 - Ordered, 15 - Trial, 30 - Active, 40 - Graced, 50 - Expired, 60 - terminated, 70 - Canceled, 80 - Suspended);
- **Int** ServStatus - parent subscription service status (10 - ''Not Provisioned'', 20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Str** StartDate - the date of the parent subscription start in Unix date format;
- **Str** ExpirationDate - expiration date of the parent subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;
- **Int** PlanID - ID of subscription service plan;
- **Int** Period - subscription period duration (for example, 1 if subscription period is 1 year);
- **Int** PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 - Month(s), 3 - Year(s);
- **Str** DomainName - domain name will be used as subscription name. If domain name is passed, it is validated according to internal rules;
- **Double** SetupFee - subscription setup fee;
- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

  Note: fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days, always passed 0;
- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);

- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>PEMGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>AddPEMSubscription_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
         <!-- startDate -->
         <value><i4>1219708800</i4></value>
         <!-- ExpirationDate -->
         <value><i4>1251244800</i4></value>
         <!-- AccountID -->
         <value><i4>1000023</i4></value>
         <!-- PlanID -->
         <value><i4>123</i4></value>
         <!-- Period -->
         <value><i4>1</i4></value>
         <!-- PeriodType -->
         <value><i4>3</i4></value>
         <!-- DomainName -->
         <value>mydomain.com</value>
         <!-- SetupFee -->
         <value><double>10</double></value>
         <!-- SubscriptionFee -->
         <value><double>12</double></value>
         <!-- RenewalFee -->
         <value><double>8</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
         <!-- NonRefundableAmt -->
         <value><double>10</double></value>
         <!-- RefundPeriod -->
         <value><i4>0</i4></value>
         <!-- BillingPeriodType -->
         <value><i4>2</i4></value>
         <!-- BillingPeriod -->
         <value><i4>1</i4></value>
         <!-- LastBillDate -->
         <value><i4>1219708800</i4></value>
         <!-- NextBillDate -->
         <value><i4>1222387200</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added subscription ID -->
              <value><i4>1037</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AddPEMSubscriptionWithID_API

Description	Parameters	
	Passed	**Returned**
Method creates new subscription based on given parameters with specified ID. Subscription information is added to PBA database and provisioning specific information (for example, resource rates) is added to PEMGATE table.	20	1

Syntax

```
ErrorMessage PEMGATE::AddPEMSubscriptionWithID_API(
  Int SubscriptionID;
  Int Status;
  Int ServStatus;
  Int StartDate;
  Int ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int Period;
  Int PeriodType;
  Str DomainName;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate.
)
returns
  Str ErrorMessage -  message after subscription has been added: "Subscription is
successfully added."
```

Special Notes

- **Int** SubscriptionID - ID of new subscription;
- **Int** Status - parent subscription status (10 - Ordered, 15 - Trial, 30 - Active, 40 - Graced, 50 - Expired, 60 - terminated, 70 - Canceled, 80 - Suspended);
- **Int** ServStatus - parent subscription service status (10 - ''Not Provisioned'', 20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Str** StartDate - the date of the parent subscription start in Unix date format;
- **Str** ExpirationDate - expiration date of the parent subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;
- **Int** PlanID - ID of subscription service plan;
- **Int** Period - subscription period duration (for example, 1 if subscription period is 1 year);
- **Int** PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 - Month(s), 3 - Year(s);
- **Str** DomainName - domain name will be used as subscription name. If domain name passed, it is validated whether subscription with the domain name is registered in POA already. If passed domain name exists in POA in *Deleted* status, it is assigned with passed subscription ID and gets to *Running* status;
- **Double** SetupFee - subscription setup fee;
- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

  Note: fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days, always passed 0;

- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);
- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>PEMGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>AddPEMSubscriptionWithID_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1005630</i4></value>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
         <!-- startDate -->
         <value><i4>1219708800</i4></value>
         <!-- ExpirationDate -->
         <value><i4>1251244800</i4></value>
         <!-- AccountID -->
         <value><i4>1000023</i4></value>
         <!-- PlanID -->
         <value><i4>123</i4></value>
         <!-- Period -->
         <value><i4>1</i4></value>
         <!-- PeriodType -->
         <value><i4>3</i4></value>
         <!-- DomainName -->
         <value>mydomain.com</value>
         <!-- SetupFee -->
         <value><double>10</double></value>
         <!-- SubscriptionFee -->
         <value><double>12</double></value>
        <!-- RenewalFee -->
         <value><double>8</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
        <!-- NonRefundableAmt -->
         <value><double>10</double></value>
         <!-- RefundPeriod -->
         <value><i4>0</i4></value>
        <!-- BillingPeriodType -->
         <value><i4>2</i4></value>
       <!-- BillingPeriod -->
         <value><i4>1</i4></value>
       <!-- LastBillDate -->
         <value><i4>1219708800</i4></value>
         <!-- NextBillDate -->
         <value><i4>1222387200</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Subscription is successfully added.
              </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AddPEMDomainForSubscription_API

	Parameters	
Description	**Passed**	**Returned**

Method adds existing domain information to respective PEMGATE table, registers domain in POA system and assigns it to specified existing POA hosting subscription. Method has no output parameters. Exception is thrown in case of POA returns some error.	2	0

Syntax

```
PEMGATE::AddPEMDomainForSubscription_API(
  Str FullDomainName;
  Int SubscriptionID.
)
```

Special Notes

- **Str** FullDomainName - domain name + TLD, without www prefix. For example, "domain.com";
- **Int** SubscriptionID.- existing POA hosting subscription ID.

Example

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>PEMGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>AddPEMDomainForSubscription_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- FullDomainName -->
         <value>mydomain.com</value>
         <!-- POA Subscription ID -->
         <value><i4>127200</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

AppliedDocsListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of the documents of selected type which are assigned to the specified document identifier.	3	8

Syntax

```
BM::AppliedDocsListGet_API(
  Int DocID;
  Int DocType;
  Int SortNo.
)
returns
  Int DocID;
  Int DocType;
  Int AdjType;
  Int DocTypeDet;
  Double AdjSum;
  Str CurrencyID;
  Int AdjustID;
  Int ReversedByAdjID.
```

Parameters Description

Input parameters:

- **Int** DocID - Identifier of the document for which you want to find the assigned documents.
- **Int** DocType - Type of the document. The type can be one of the following:
  - `1` - the type is *Order.*
  - `2` - the type is *Document*
- **Int** SortNo - Parameter defining how the output data is sorted. The parameter is set the following way:
  - `1` - The data is sorted by the first column in ascending order.
  - `2` - The data is sorted by the second column in ascending order.
  - `-1` - The data is sorted by the first column in descending order.
  - `-2` - The data is sorted by the second column in descending order.

Output parameters:

- **Int** DocID - Identifier of the document for which you want to find the assigned documents.
- **Int** DocType - Type of the document. The type can be one of the following:
    - `1` - the type is *Order.*
    - `2` - the type is *Document*
- **Int** AdjType - Parameter defining the type of relation between the document specified in the input data and the document specified in the output data. The value of this parameter can be as follows:
    - `100` - Adjustment
    - `200` - Reference
- **Int** DocTypeDet - Details of the type to which the document belongs. The value of this parameter depends on the following:
    - If the document is an invoice or payment (*ARdoc* class), the values can be:
        - `20` - Invoice
        - `50` - Payment
        - `70` - Refund
        - `75` - Void Check
        - `80` - Credit Memo
        - `90` - Debit Memo
        - `100` - Fraud Check
    - If the document is an order, the parameter value is `0`.
- **Double** AdjSum - Adjustment sum.
- **Str** CurrencyID - Identifier of the currency used to perform payments.
- **Int** AdjustID - Internal ID of the document applying operation. Indicates that the listed document was applied to the document specified in the input data.
- **Int** ReversedByAdjID - The parameter passes 0 in case the original apllying is preserved, or the internal ID of the new applying operation if the original one was reversed.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>AppliedDocsListGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
            <!-- Doc ID -->
           <value><i4>76</i4></value>
            <!-- Doc Type -->
           <value><i4>2</i4></value>
            <!-- Sort No -->
           <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                 <!-- Doc ID -->
```

```
                <value><i4>77</i4></value>
                 <!-- Doc Type -->
                <value><i4>2</i4></value>
                 <!-- Adj Type -->
                <value><i4>100</i4></value>
                 <!-- ARDoc Type -->
                <value><i4>20</i4></value>
                 <!-- Adjustment Amount -->
                <value><double>20.000000</double></value>
                 <!-- Currency ID -->
                <value><string>USD</string></value>
                 <!-- Adjust ID -->
                     <value><i4>1002</i4></value>
                 <!-- Reversed By AdjID -->
                     <value><i4>0</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

AppliedDocsListGetEx_API

Description	Parameters	
	Passed	Returned
Method returns the list of the documents of selected type which are assigned to the specified document identifier. Comparing to the AppliedDocsListGet_API (on page 115) method, there is the option to enable listing of the document applying operations.	4	8

Syntax

```
BM::AppliedDocsListGetEx_API(
  Int DocID;
  Int DocType;
  Int IncludeReversedTrans;
  Int SortNo.
)
returns
  Int DocID;
  Int DocType;
  Int AdjType;
  Int DocTypeDet;
  Double AdjSum;
  Str CurrencyID;
  Int AdjustID;
  Int ReversedByAdjID.
```

Parameters Description

Input parameters:

- **Int** DocID - Identifier of the document for which you want to find the assigned documents.
- **Int** DocType - Type of the document. The type can be one of the following:
  - `1` - the type is *Order.*
  - `2` - the type is *Document*
- **Int** IncludeReversedTrans - Parameter specifies whether the information on reversed document applying operations should be displayed.
  - `1` - Reversed document applying operations are displayed.
  - `0` - Reversed document applying operations are not displayed.
- **Int** SortNo - Parameter defining how the output data is sorted. The parameter is set the following way:
  - `1` - The data is sorted by the first column in ascending order.
  - `2` - The data is sorted by the second column in ascending order.
  - `-1` - The data is sorted by the first column in descending order.
  - `-2` - The data is sorted by the second column in descending order.

Output parameters:

- **Int** DocID - Identifier of the document for which you want to find the assigned documents.
- **Int** DocType - Type of the document. The type can be one of the following:
  - `1` - the type is *Order.*
  - `2` - the type is *Document*
- **Int** AdjType - Parameter defining the type of relation between the document specified in the input data and the document specified in the output data. The value of this parameter can be as follows:
  - `100` - Adjustment
  - `200` - Reference
- **Int** DocTypeDet - Details of the type to which the document belongs. The value of this parameter depends on the following:
  - If the document is an invoice or payment (*ARdoc* class), the values can be:

    - `20` - Invoice
    - `50` - Payment
    - `70` - Refund
    - `75` - Void Check
    - `80` - Credit Memo
    - `90` - Debit Memo
    - `100` - Fraud Check
  - If the document is an order, the parameter value is `0`.
- **Double** AdjSum - Adjustment sum.
- **Str** CurrencyID - Identifier of the currency used to perform payments.
- **Int** AdjustID - Internal ID of the document applying operation. Indicates that the listed document was applied to the document specified in the input data.
- **Int** ReversedByAdjID - The parameter passes 0 in case the original apllying is preserved, or the internal ID of the new applying operation if the original one was reversed.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>AppliedDocsListGetEx_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
            <!-- Doc ID -->
           <value><i4>76</i4></value>
            <!-- Doc Type -->
           <value><i4>2</i4></value>
            <!-- IncludeReversedTrans -->
           <value><i4>1</i4></value>
            <!-- Sort No -->
           <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
```

```
               <data>
                 <!-- Doc ID -->
                <value><i4>77</i4></value>
                 <!-- Doc Type -->
                <value><i4>2</i4></value>
                 <!-- Adj Type -->
                <value><i4>100</i4></value>
                 <!-- ARDoc Type -->
                <value><i4>20</i4></value>
                 <!-- Adjustment Amout -->
                <value><double>20.000000</double></value>
                 <!-- Currency ID -->
                <value><string>USD</string></value>
                 <!-- Adjust ID -->
                     <value><i4>1002</i4></value>
                 <!-- Reversed By AdjID -->
                     <value><i4>0</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodResponse>
```

ARDocDetailsListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the details of the invoice associated with the specified identifier (*DocID*). **Note:** in the results returned, the pricing matches the Extra Precision configuration in PBA. For more details refer to the *Extra Precision in PBA Pricing* section in *PBA Provider's Guide.*	2	32

Syntax

```
BM::ARDocDetailsListGet_API(
  Int DocID;
  Int SortNo.
)
returns
  Int DetID;
  Int DocID;
  Int DetType;
  Str SKU;
  Str Descr;
  Double UnitPrice;
  Double Quantity;
  Str MeasureUnit;
  Double ServQty;
  Str ServUnitMeasure;
  Double ExtendedPrice;
  Int DurBillPeriod;
  Int DurBillPeriodType;
  Double Duration;
  Int PlanPeriodID;
  Int PromoID;
  Int DiscID;
  Int resourceID;
  Double DiscountAmt;
  Str TaxCatID;
  Double TaxAmt;
  Double TaxInclAmt;
  Int ResCatID;
  Int planCategoryID;
  Int OIID;
  Int DDOrdDetID;
  Int DDOrderID;
  Int subscriptionID;
  Int DetSDate;
  Int DetEDate;
  Int DetBaseDate;
Int TaxAlg.
```

Parameters Description

Input parameters:

- **Int** DocID - Identifier of the document the details for which are to be displayed.
- **Int** SortNo - Parameter defining how the output data is sorted. The parameter is set the following way:
    - `1` - The data is sorted by the first column in ascending order.
    - `2` - The data is sorted by the second column in ascending order.
    - `-1` - The data is sorted by the first column in descending order.
    - `-2` - The data is sorted by the second column in descending order.

Output parameters:

- **Int** DetID - Line identifier.
- **Int** DocID - Identifier of the document the details for which are to be displayed.
- **Int** DetType - Document detail type. The type can be as follows:
    - `0` - Unknown
    - `100` - Plan Setup
    - `101` - Plan Switch
    - `110` - Plan Subscription
    - `115` - Plan Transfer
    - `120` - Resource Setup
    - `125` - Resource Switch
    - `130` - Resource Subscription
    - `140` - Resource overuse
    - `150` - Resource Downgrade
    - `400` - Discount
    - `600` - Adjustment
    - `650` - Cancellation
    - `699` - Bill Penalty
    - `700` - Non Provisioning Item
    - `799` - Bill Record
    - `800` - Payment
    - `900` - Refund

  - `905` - Refund Adjustment
  - `910` - Void Check
  - `10100` - Plan Setup Refund
  - `10110` - Plan Subscription Refund
  - `10120` - Resource Setup Refund
  - `10130` - Resource Subscription Refund
  - `11510` - Plan Transfer Refund
- **Str** SKU - Name of the SKU (Stock Keeping Unit) of the purchased item.
- **Str** Descr - Document description.
- **Double** UnitPrice - Price per resource unit.
- **Double** Quantity - This field is obsolete. See the description of the `ServQty` parameter.
- **Str** MeasureUnit - This field is obsolete. See the description of the `ServUnitMeasure` parameter.
- **Double** ServQty - Quantity of the purchased services.
- **Str** ServUnitMeasure - Unit of measure for services.
- **Double** ExtendedPrice - Total sum charged for the item.
- **Int** DurBillPeriod - Duration of the billing period.
- **Int** DurBillPeriodType - Type of the billing period duration.
- **Double** Duration - Value displayed in the **Duration** column of the order details list.
- **Int** PlanPeriodID - Identifier of the subscription period applied to the order detail.
- **Int** PromoID - Identifier of the promotion type that is applied to the order detail.
- **Int** DiscID - Identifier of a discount applied to the order detail.
- **Int** resourceID - Identifier of the resource purchased.
- **Double** DiscountAmt - Total sum of the discount provided.
- **Str** TaxCatID - Identifier of the tax category applicable for the purchased item.
- **Double** TaxAmt - Total amount of tax.
- **Double** TaxInclAmt - Total amount charged for the purchase including tax.
- **Int** ResCatID - Identifier of the resource category applied to the order detail.
- **Int** planCategoryID - Identifier of the plan category applied to the order detail.
- **Int** OIID - Order identifier.
- **Int** DDOrdDetID - Link to the order detail.
- **Int** DDOrderID - Link to the order.
- **Int** subscriptionID - Identifier of the subscription.

- **Int** DetSDate - Detail start date.
- **Int** DetEDate - Detail end date.
- **Int** DetBaseDate - Abstract redundant value on the base of which the `DetSDate` and `DetEDate` parameters can be calculated.
- **Int** TaxAlg - Applied taxation algorithm. The value of this parameter can be one of the following:
    - `0` - Regular taxation is applied.
    - `10` - Manual taxation is applied.

Note: This method can return the 3.4028234663852886e+38 value, which means 0.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>ARDocDetailsListGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <value><i4>2</i4></value>
           <value><i4>2</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
```

```
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <value><i4>60</i4></value>
                 <value><i4>59</i4></value>
                 <value><i4>100</i4></value>
                 <value><string></string></value>
                 <value><string>Shared Hosting Setup</string></value>
                 <value><double>10.000000</double></value>
                 <value><double>1.000000</double></value>
                 <value><string>item</string></value>
                 <value><double>1.000000</double></value>
                 <value><string>item</string></value>
                 <value><double>10.000000</double></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>-2147483648</i4></value>

<value><double>340282346638528859811704183484516925440.000000</double></value>
                 <value><i4>1</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><double>0.000000</double></value>
                 <value><string>TaxCat</string></value>
                 <value><double>0.000000</double></value>
                 <value><double>0.000000</double></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>1</i4></value>
                 <value><i4>86</i4></value>
                 <value><i4>44</i4></value>
                 <value><i4>107</i4></value>
                 <value><i4>1000015</i4></value>
                 <value><i4>1363035600</i4></value>
                 <value><i4>1363035600</i4></value>
                 <value><i4>1363035600</i4></value>
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
              <value>
               <array>
                <data>
                 <value><i4>61</i4></value>
                 <value><i4>59</i4></value>
                 <value><i4>110</i4></value>
                 <value><string></string></value>
                 <value><string>Shared Hosting Recurring</string></value>
                 <value><double>40.000000</double></value>
                 <value><double>1.000000</double></value>
                 <value><string>month(s)</string></value>
                 <value><double>1.000000</double></value>
                 <value><string>item</string></value>
                 <value><double>40.000000</double></value>
                 <value><i4>1</i4></value>
                 <value><i4>4</i4></value>
                 <value><double>1.000000</double></value>
                 <value><i4>1</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>-2147483648</i4></value>
                 <value><double>0.000000</double></value>
                 <value><string>TaxCat</string></value>
                 <value><double>0.000000</double></value>
                 <value><double>0.000000</double></value>
                 <value><i4>-2147483648</i4></value>
                 <value><i4>1</i4></value>
                 <value><i4>86</i4></value>
                 <value><i4>45</i4></value>
                 <value><i4>107</i4></value>
                 <value><i4>1000015</i4></value>
                 <value><i4>1363035600</i4></value>
                 <value><i4>1365710400</i4></value>
                 <value><i4>1363035600</i4></value>
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ARDocInfoGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the details of the specified invoice.	1	18

Syntax

```
BM::ARDocInfoGet_API(
  Int InvoiceID.
)
returns
Int DocID;
Str DocNum;
Int DocType;
Int Status;
Str CurrencyID;
Double Total;
Double TaxTotal;
Double DocBBalance;
Double AvailBal;
Int Vendor_AccountID;
Int Customer_AccountID;
Int Crtd_DateTime;
Int Crtd_User_UsersID;
Int DocDate;
Int DueDate;
Str Description;
Str BranchID;
Str SalesID.
```

Parameters Description

Input parameters:

Int InvoiceID - Identifier of the invoice the details of which are to be displayed.

Output parameters:

- **Int** DocID - Internal identifier of the document.
- **Str** DocNum - Number of the document.
- **Int** DocType - Type of the document. The type can be one of the following:
  - `20` - Invoice
  - `50` - Payment
  - `70` - Refund

    - `75` - Void Check
    - `80` - Credit Memo
    - `90` - Debit Memo
    - `100` - Fraud Check
- **Int** Status - Status of the document. The status can be one of the following:
    - `1000` - Open
    - `2000` - Hold
    - `3000` - Closed
    - `4000` - Voided
    - `5000` - Refunded
    - `6000` - Cancelled
    - `7000` - Delayed
- **Str** CurrencyID - Currency used for payments.
- **Double** Total - *Total Sum* charged for the purchase specified in the document.
- **Double** TaxTotal - *Total Tax* sum specified in the document.
- **Double** DocBBalance - *Accounting Balance* specified in the document.
- **Double** AvailBal - *Available Balance* specified in the document.
- **Int** Vendor_AccountID - Identifier of the related vendor account.
- **Int** Customer_AccountID - Identifier of the customer account.
- **Int** Crtd_DateTime - Document creation date and time.
- **Int** Crtd_User_UsersID - Identifier of the user who has created the document.
- **Int** DocDate - Date of the document issuing.
- **Int** DueDate - Date by which the purchase must be paid.
- **Str** Description - Description of the document.
- **Str** BranchID - Identifier of the sales branch associated with the customer account.
- **Str** SalesID - Identifier of the sales person associated with the customer account.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>ARDocInfoGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- InvoiceID-->
           <value><i4>2</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <!-- DocID-->
             <value><i4>59</i4></value>
             <!-- DocNum-->
             <value><string>000033</string></value>
             <!-- DocType-->
```

```
                <value><i4>20</i4></value>
                <!-- Status-->
                <value><i4>3000</i4></value>
                <!-- CurrencyID-->
                <value><string>USD</string></value>
                <!-- Total-->
                <value><double>50.000000</double></value>
                <!-- TaxTotal-->
                <value><double>0.000000</double></value>
                <!-- DocBBalance-->
                <value><double>0.000000</double></value>
                <!-- AvailBal-->
                <value><double>0.000000</double></value>
                <!-- Vendor AccountID-->
                <value><i4>1</i4></value>
                <!-- Customer AccountID-->
                <value><i4>1000002</i4></value>
                <!-- Crtd  DateTime-->
                <value><i4>1363081048</i4></value>
                <!-- User  UsersID-->
                <value><i4>-1</i4></value>
                <!-- DocDate-->
                <value><i4>1363035600</i4></value>
                <!-- DueDate-->
                <value><i4>1363899600</i4></value>
                <!-- Description-->
                <value><string>Order for Subscription on Plan #1 (Shared Hosting)
for 1 Year(s). </string></value>
                <!-- BranchID-->
                <value><string>888</string></value>
                <!-- SalesID-->
                <value><string>123</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </member>
        </struct>
       </value>
      </param>
     </params>
    </methodResponse>
```

AttrTypeByVendorListGet_API

Description	Parameters	
	Passed	Returned

This method returns the list of attributes assigned to a specified vendor. **Note**: If an identifier of a customer account is specified instead of a vendor account one, the method returns all the attributes of an hierarchy (a provider, a reseller, a sub-reseller, and so on) to which the specified customer belongs.	2	The number of returned parameters depends on the number of assigned attributes.

Syntax

```
BM::AttrTypeByVendorListGet_API(
  Int Vendor;
  Int SortNo.
)
returns
  an array of items containing attribute data:
  Str AttributeID;
  Str Name;
  Str Description;
  Str Type;
  Str Group;
  Int Status;
  Int AfterField;
  Int AttributeType;
  Int Active;
  Int SortNumber;
  Int ShowInStore;
  Str Validator;
  Str Comment;
  Int UniqueSetting;
  Str Tags;
  Int VendorID.
```

Parameters Description

Input parameters:

- **Int** VendorID - The identifier of the vendor for whom the assigned attributes are to be listed.
- **Int** SortNo - This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, N is the number of a column):
  - N - The data is sorted by the N column in ascending order.
  - - N - The data is sorted by the N column in descending order.

Output parameters:

An array of items that contain the following attribute data:

- **Str** AttributeID - Identifier of an attribute assigned to the specified vendor.

- **Str** Name - Name of the attribute.
- **Str** Description - Description of the attribute.
- **Str** Type - Type of the attribute data. The parameter value can be as follows:
  - *INT* - an attribute of this type prompts a customer to enter an integer value.
  - *STR* - an attribute of this type prompts a customer to enter a text up to 80 characters in length. The input box is displayed as a single line in this case.
  - *STRA* - an attribute of this type prompts a customer to enter a text over 80 characters in length. The input box is displayed as a text box.
  - *DATE* - an attribute of this type prompts a customer to enter a date. The calendar icon is displayed next to the input box.
  - *PASSWD* - an attribute of this type prompts a customer to enter a password.
  - *NAME_LOGIN* - an attribute of this type prompts a customer to enter a login name.
  - *DOMAIN_NAME* - an attribute of this type prompts a customer to enter a domain name.
  - *STRASCII* - an attribute of this type prompts a customer to enter a text in the ASCII encoding. It is useful when entering the contact information: strictly the ASCII format is required.
- **Str** Group - Name of the group to which the attribute belongs.
- **Int** Status - This parameter defines whether specifying the attribute value is to be mandatory or optional. The parameter value can be one of the following:
  - *10* - -if entering the attribute value is optional.
  - *20* - -if entering the attribute value is required.
- **Int** AfterField - This parameter defines a field after which the attribute will be shown in the adding new entity form. The parameter is shown only in the online store.
- **Int** AttributeType - This parameter defines to what PBA entity the attribute is applicable. The parameter value can be as follows:
  - *Accounts* - the attribute is shown only in the customer registration form.
  - *Users* - the attribute is shown only in the provider users registration form.
  - *Order* - the attribute is shown on placing an order.
- **Int** Active - The parameter defines whether the attribute is active or not. The parameter value can be as follows:
  - *1* - if the attribute is active.
  - *0* - if the attribute is not active.
- **Int** SortNumber - This is an optional parameter defining by which column the output data is sorted.
- **Int** ShowInStore - This parameter defines whether to display the attribute in the online store or to do in only on creating an account from CPP:

    - *1* - if the attribute is shown in the storefront.
    - *0* - if the attribute is not shown in the storefront.
- **Str** Validator - Regular expression that is used to check the attribute value validity.
- **Str** Comment - Short description of the attribute.
- **Int** UniqueSetting - This option is applicable only to the *accounts* type attributes. It shows the type of uniqueness of the attribute:
    - *Not unique* – Uniqueness is not necessary for the attribute.
    - *Unique per Vendor* – The attribute must be unique within a vendor system.
    - *Unique globally* – The attribute must be totally unique.
- **Str** Tags - This parameters shows tags that are assigned to the attribute.
- **Int** VendorID - Identifier of the vendor for whom the assigned attributes are to be listed.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the xml comments. REMOVE all the comments from an actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>AttrTypeByVendorListGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
            <!-- Identifier of the vendor for which you want to get the attributes list displayed. -->
                  <value>
                    <i4>1000005</i4>
                  </value>
           <!-- Number of column by which you want the output data to be sorted. -->
                  <value>
                    <i4>1</i4>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
```

```
            <value>
             <array>
              <data>
<!-- Identifier of an attribute. -->
               <value><string>CLE</string></value>
<!-- Name of the attribute. -->
               <value><string>CLE</string></value>
<!-- Description of the attribute -->
               <value><string>Client Legal Entity</string></value>
<!-- Type of the attribute data. -->
               <value><string>STR</string></value>
<!-- Name of the attributes group to which the attribute belongs. -->
               <value><string>Additional Account Information<string></value>
<!-- This parameter defines whether specifying the attribute value is to be mandatory (20) or optional (10). -->
               <value><i4>10</i4></value>
<!-- This parameter defines a field after which the attribute will be shown in the adding new entity form. -->
               <value><i4>Address Line 1</i4></value>
<!-- This parameter defines to what PBA entity (an account, user, or order ) the attribute is applicable. -->
               <value><i4>Accounts</i4></value>
<!-- The parameter defines whether the attribute is active (1) or not (0). -->
               <value><i4>1</i4></value>
<!-- This is an optional parameter defining by which column the output data is sorted. -->
               <value><i4>1</i4></value>
<!-- This parameter defines whether to display the attribute in the online store (1) or to do in only on creating an account from CPP (0).
-->
               <value><i4>1<i4></value>
<!-- Regular expression that is used to check the attribute value validity. -->
               <value><string> ^[0-9]{1,15}$ </string></value>
<!-- A short explanation that will be shown to a customer when inputing the attribute value. For instance, you can enter an example of
the attribute value demonstrating how to input it in the correct format. -->
               <value><string> 15-(or less) characters number.</string></value>
<!-- This option is applicable only to the accounts type attributes. It shows the type of uniqueness of the attribute. -->
               <value><i4>Unique globally</i4></value>
<!-- This parameters shows tags that are assigned to the attribute. -->
               <value><string>COMP, PERS</string></value>
<!-- Identifier of the vendor to which the attribute is assigned. -->
               <value><i4>1000005</i4></value>
              </data>
             </array>
            </value>
            <value>
             <array>
              <data>
               <value><string>CNPJ</string></value>
               <value><string>CNPJ</string></value>
               <value><string>Brazilian tax payer's identifier for
companies.</string></value>
               <value><string>STR</string></value>
               <value><string>Additional Account Information</string></value>
               <value><i4>20</i4></value>
               <value><i4>Address Line 1</i4></value>
               <value><i4>Accounts</i4></value>
               <value><i4>1</i4></value>
               <value><i4>1</i4></value>
               <value><i4>1</i4></value>

<value><string>^\d{2,2}\.\d{3,3}\.\d{3,3}\/\d{4,4}\\d{2,2}$|^\d{14,14}$
</string></value>
               <value><string>##.###.###/####-## </string></value>
               <value><i4> Unique pre vendor</i4></value>
               <value><string>COMP</string></value>
               <value><i4>1000005</i4></value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

AttrTypeListGet_API

Description	Parameters	
	Passed	Returned
Method returns list of attributes with details registered in PBA.	1	10

Syntax

```
ListResult BM::AttrTypeListGet_API(
  Int SortNo.
)
returns
 Str AttributeID - ID of attribute;
 Str Name - attribute name;
 Str Description - attribute description;
 Str Type - attribute value type. INT - integer, STR - text up to 80 characters,
STRA - text over 80 characters, DATE - date, PASSWD - password, NAME_LOGIN -
login name, DOMAIN_NAME - domain name;
 Str Group - attribute group name. Attributes with the same group displayed in
online store under group name;
 Int Status - defines whether attribute is optional - 10, or required - 20;
 Int AfterField - defines where the attribute should be located among other data
fields which a customer has to fill in: 1 - Address Line 1, 2 - Address Line 2, 3
- City, 4 - Country, 5 - Email Address, 6 - Phone, 7 - Fax, 8 - First Name, 9 -
Last Name, 10 - Company, 11 - Postal address, 12 - Tax, 13 - State/Province, 14 -
Zip code, 15 - Login;
 Int AttributeType - defines attribute type: 0 - available for accounts, 1 -
available for users, 2 - available for orders;
 Int Active - defines whether attribute is active - 1 or not - 0;
 Int SortNumber - attribute sort number. It works together with the After Field
option. If a number of attributes have same the After Field parameter value, the
parameter defines sort number for them.
 Int ShowinStorefront - signifies whether to display an attribute in online store
-1 or only on creating account from CP - 0;
 Str Validator - the regular expression against which the value of attribute will
be validated;
 Str Comment - the attribute's hint that is displayed on screen;
 Int UniqueSetting - specifies whether the attribute value should be unique: 0 -
Not unique, 10 - Unique per Vendor, 20 - Unique globally;
 Str Tags - attribute tags: COMP, PERS are system predefined tags and refer to the
company and personal accounts, respectively;
 Int VendorID - ID of the vendor under which the attribute was created;
 Int DataTypeID - unique ID of the attribute's data type.
```

Special Notes

Int SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Attribute ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>AttrTypeListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
```

```
<!-- Attribute ID -->
 <value>
        <string>CompTaxPayerID</string>
 </value>
<!-- Attribute Name -->
<value>
        <string>INN</string>
 </value>
<!-- Attribute Description -->
<value>
        <string/>
 </value>
<!-- Attribute Value Type -->
<value>
        <string>STR</string>
 </value>
<!-- Group -->
<value>
        <string>Attributes of companies</string>
</value>
<!-- Status -->
<value>
        <i4>20</i4>
</value>
<!-- After Field -->
<value>
        <i4>1</i4>
</value>
<!-- Attribute Type -->
<value>
        <i4>0</i4>
</value>
<!-- Active -->
<value>
        <i4>1</i4>
</value>
<!-- Sort Number -->
<value>
        <i4>-2147483648</i4>
</value>
<!-- ShowinStorefront -->
<value>
        <i4>1</i4>
</value>
<!-- Validator -->
<value>
        <string>^\d{10,10}$|^\d{12,12}$</string>
</value>
<!-- Comment -->
<value>
        <string>should be 10 or 12 digit number</string>
</value>
<!-- UniqueSetting -->
<value>
        <i4>10</i4>
</value>
<!-- Tags -->
<value>
        <string>COMP</string>
</value>
```

```
                                        <!-- Vendor ID -->
                                        <value>
                                                    <i4>1</i4>
                                        </value>
                                        <!-- DataTypeID -->
                                        <value>
                                                    <i4>1</i4>
                                        </value>
                                     </data>
                                   </array>
                                 </value>
                               </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

BankAccountAdd_API

Description	Parameters	
	Passed	**Returned**
The method creates a *BankAccount* payment method for the specified account.	7	1

Syntax

```
ItemResult BM::BankAccountAdd_API (
  Int AccountID;
  Str PaySystemID;
  Str BankNumber;
  Str AccountNumber;
  Str AccountHolderName;
  Str RegisteredPlace;
  Int UseForAutoPayments;
)
returns
  Int PayToolID.
```

Parameters Description

Input Parameters:

- **Int** AccountID – Identifier of the PBA account for which the *BankAccount* payment method is to be created.
- **Str** PaySystemID – Identifier of the corresponding payment system in PBA.
- **Str** BankNumber – Number of the bank in which an account that is to be used for payments is registered.
- **Str** AccountNumber – Number of the customer account in this bank.
- **Str** AccountHolderName – Full name of the bank account holder.
- **Str** RegisteredPlace – Depending on the system that uses the bank account, a name or a code of the place where the bank account was registered.
- **Int** UseForAutoPayments – Parameter defining whether to use the *BankAccount* payment method for autopayments for the specified account. The parameter value can be:
  - 0 if the method is not to be used for autopayments.
  - 1 if the method is to be used for autopayments.

Output Parameter:

Int PayToolID – Identifier of the created payment method.

Example

Request

```
<?xml version="1.0" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse XML comments, so REMOVE all the comments from an actual request. -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>AutoCommit</name>
            <value>Yes</value>
          </member>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>BankAccountAdd_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
             <array>
              <data>
               <!-- AccountID -->
               <value><i4>1000001</i4></value>
               <!-- PaySystemID -->
               <value><string>Telstra Payment System</string></value>
               <!-- BankNumber -->
               <value><string>Bank Num #1</string></value>
               <!-- AccountNumber -->
               <value><string>10101010101</string></value>
               <!-- AccountHolderName -->
               <value><string>John Doe</string></value>
               <!-- RegisteredPlace -->
               <value><string>Neverland</string></value>
               <!-- UseForAutoPayments -->
               <value><i4>0</i4></value>
              </data>
             </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
```

```
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <!-- PayToolID -->
            <value><i4>8</i4></value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

BankAccountGet_API

Description	Parameters	
	Passed	Returned
This method returns information about the bank account associated with the specified payment method of the *bank account* type.	1	3

Syntax

```
BM::BankAccountGet_API(
  Int PayToolID.
)
returns
  Str AccountNumber;
  Str BankNumber;
  Str AccHolderName.
```

Parameters Description

Input parameter:

Int PayToolID - Identifier of a customer payment method.

Output parameters:

- **Str** AccountNumber - Number of the customer bank account, which is used for payment.
- **Str** BankNumber - Number of the bank to which the account belongs.
- **Str** AccHolderName - Name of the bank account holder.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>BankAccountGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value><string>1111222233334444</string></value>
              <value><string>987</string></value>
              <value><string>JOHN SMITH</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

BannerRedirectURLGet_API

Description	Parameters	
	Passed	Returned
Method returns redirect link for specific marketing campaign.	1	1

Syntax

```
ItemResult BM::BannerRedirectURLGet_API(
  Int BannerID.
)
returns
  Str(250) RedirectURL - redirect URL.
```

Special Notes

Int BannerID - campaign ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>BannerRedirectURLGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- BannerID -->
         <value><i4>100001</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Redirect URL -->
              <value><string>
               https://www.yoursite.com/index.php?redirect=win_standard
              </string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

BlobDocAdd_API

Description	Parameters	
	Passed	Returned
Method adds record to the BlobDocument table.	3	1

Syntax

```
ItemResult BLOBDOC::BlobDocAdd_API(
  Str Name;
  Int CompressType;
  Base64 Content;
)
returns
  Int DocID - ID of the added record.
```

Special Notes

- **Str** Name - uploaded file name without extension;
- **Int** CompressType - uploaded file type. Possible values: 0 - binary data, 1 - text, 2 - GIF, 3 - JPEG, 4 - PNG;
- **Base64** Content - base64 coded file content.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>BLOBDOC</value>
      </member>
       <member>
      <name>Method</name>
      <value>BlobDocAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Name -->
         <value>16_delete</value>
         <!-- CompressType -->
         <value><i4>2</i4></value>
         <!-- Content -->
         <value><base64>R0lGODlhEAAQAKIAANwAAP8AANFAQO1/f////
                        wAAAAAAAAAAACH5BAEAAAQALAAAAAAAQA
                        BAAAAMpSLrc/jA2ACQDoTq9cJ4P9RGUhFFcdFpk
                        Vqrju41K2gXBAG62IrBASQIAOw==
           </base64></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
```

```
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added Document ID -->
              <value><i4>10014</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

BlobDocumentMoveToArc_API

Description	Parameters	
	Passed	**Returned**
Method removes specified record from the BlobDocument table.	1	1

Syntax

```
ErrorMessage BLOBDOC::BlobDocumentMoveToArc_API(
  Int DocID;
)
returns
  Str ErrorMessage - message after specified record has been removed: "Data have
been deleted."
```

Special Notes

Int DocID - ID of record to be removed;

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
```

```
     <member>
      <name>Server</name>
       <value>BLOBDOC</value>
      </member>
       <member>
      <name>Method</name>
      <value>BlobDocumentMoveToArc_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- DocID -->
         <value><i4>10014</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Data have been deleted.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

BlobDocUpdate_API

Description	Parameters	
	Passed	**Returned**
Method updates specified record in the BlobDocument table.	4	1

Syntax

```
ErrorMessage BLOBDOC::BlobDocUpdate_API(
  Int DocID
  Str Name;
  Int CompressType;
  Base64 Content;
)
returns
  Str ErrorMessage - message after specified record has been updated: "Operation
done."
```

Special Notes

- **Int** DocID - ID of record to be updated;
- **Str** Name - uploaded file name without extension;
- **Int** CompressType - uploaded file type. Possible values: 0 - binary data, 1 - text, 2 - GIF, 3 - JPEG, 4 - PNG;
- **Base64** Content - base64 coded file content.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>BLOBDOC</value>
      </member>
       <member>
      <name>Method</name>
      <value>BlobDocAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- DocID -->
         <value><i4>10015</i4></value>
         <!-- Name -->
         <value>16_delete</value>
         <!-- CompressType -->
         <value><i4>2</i4></value>
         <!-- Content -->
         <value><base64>R0lGODlhEAAQAKIAANwAAP8AANFAQO1/f////
                        wAAAAAAAAAAACH5BAEAAAQALAAAAAAAQA
                        BAAAAMpSLrc/jA2ACQDoTq9cJ4P9RGUhFFcdFpk
                        Vqrju41K2gXBAG62IrBASQIAOw==
           </base64></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
```

```
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Operation done.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

TempBlobDocAdd_API

Description	Parameters	
	Passed	Returned
Method adds record to the BlobDocument table for specified period of time. The record is removed when specified period ends.	4	1

Syntax

```
ItemResult BLOBDOC::TempBlobDocAdd_API(
  Str Name;
  Int CompressType;
  Base64 Content;
  Int Period;
)
returns
  Int DocID - ID of the added record.
```

Special Notes

- **Str** Name - uploaded file name without extension;
- **Int** CompressType - uploaded file type. Possible values: 0 - binary data, 1 - text, 2 - GIF, 3 - JPEG, 4 - PNG;
- **Base64** Content - base64 coded file content;
- **Int** Period - period of time in seconds.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>BLOBDOC</value>
      </member>
       <member>
      <name>Method</name>
      <value>BlobDocAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Name -->
         <value>16_delete</value>
         <!-- CompressType -->
         <value><i4>2</i4></value>
         <!-- Content -->
         <value><base64>R0lGODlhEAAQAKIAANwAAP8AANFAQO1/f////
                        wAAAAAAAAAAACH5BAEAAAQALAAAAAAAQA
                        BAAAAMpSLrc/jA2ACQDoTq9cJ4P9RGUhFFcdFpk
                        Vqrju41K2gXBAG62IrBASQIAOw==
           </base64></value>
          <!-- Period -->
         <value><i4>200000</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
```

```
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added Document ID -->
              <value><i4>10014</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

CancelAllSubscriptionsForAccount_API

	Parameters	
Description	**Passed**	**Returned**
This method places and provisions the cancelation orders for those subscriptions of a specified account, which are in the *Not Provisioned*, *Stopped* or *Running* status.	4	1

Syntax

```
ListResult BM :: CancelAllSubscriptionsForAccount_API (
 Int AccountID,
 Int CancelType,
 Int ReasonID,
 Str ReasonDescription.
)
returns
 Str ErrorMessage.
```

Parameters Description

Input Parameters:

- **Int** AccountID – Identifier of a customer account, the subscriptions of which are to be processed.
- **Int** CancelType – Type of subscription cancelation. The types and their meanings are as follows:

    - 10 – the customer is refunded.
    - 20 – the customer account is credited (a credit memo is created).
    - 30 – the customer is not refunded/credited.
- **Int** ReasonID – Identifier of a system reason code, which explains the reason for the subscriptions cancelation. You can find a description of reason codes in the Provider Control Panel at **System > Settings > Order Processing > Reason Codes**.
- **Str** ReasonDescription – Optional comment explaining the reason of subscriptions cancelation in more detail.

Output Parameter:

Str ErrorMessage – Message issued after the subscriptions of the specified account have been canceled. The message content is the following: "Operation done."

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>CancelAllSubscriptionsForAccount_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
             <!-- AccountID -->
             <value><i4>1000003</i4></value>
                    <!-- CancelType -->
             <value><i4>30</i4></value>
                    <!-- ReasonID -->
             <value><i4>14</i4></value>
                    <!-- ReasonDescription -->
             <value><string>Cancelation due to valid business
reasons</string></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done.</string></value>
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

ChangeParentSubscription_API

Description	Parameters	
	Passed	**Returned**
The method changes the parent subscription for a specified subscription.	2	A message informing about the operation result.

Syntax

```
ItemResult BM::ChangeParentSubscription_API(
  Int SubscriptionID;
  Int ParentSubscriptionID.
)
returns
  Int Message.
```

Parameters Description

Input Parameters:

- **Int** SubscriptionID – Identifier of the subscription for which the parent subscription is to be changed.
- **Int** ParentSubscriptionID – Identifier of a new parent subscription for the specified subscription.

Output Parameter:

- **Int** Message– A message informing about the operation result having the following content: "Operation done."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request. -->
    <methodCall>
      <methodName>Execute</methodName>
        <params>
          <param>
            <value>
              <struct>
                <member>
                  <name>Server</name>
                  <value>BM</value>
                </member>
                <member>
                   <name>Method</name>
                   <value>ChangeParentSubscription_API</value>
                </member>
                <member>
                   <name>Params</name>
                   <value>
                    <array>
                      <data>
                        <!-- SubscriptionID. -->
                        <value><int>1000003</int></value>
                        <!-- ParentSubscriptionID. -->
                        <value><int>1000001</int></value>
                      </data>
                    </array>
                   </value>
                </member>
              </struct>
            </value>
          </param>
       </params>
     </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <struct>
            <member>
             <name>Status</name>
             <!-- Message -->
             <value><string>Parent Subscription has been changed.
</string></value>
```

```
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

ChildSubscriptionListGet_API

	Parameters	
Description	**Passed**	**Returned**
The method returns the list of child subscriptions that the specified subscription has.	2	a list of child subscription IDs

Syntax

```
ItemResult BM::ChildSubscriptionListGet_API(
  Int SubscriptionID;
  Int SortNo.
)
returns
  Int ChildSubscriptionID.
```

Parameters Description

Input Parameters:

- **Int** SubscriptionID – Identifier of the subscription for which the child subscriptions are to be found.
- **Int** SortNo – This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, N is the number of a column):
  - N - The data is sorted by the N column in ascending order.
  - - N - The data is sorted by the N column in descending order.

Output Parameter:

A list of identifiers of the child subscriptions:

- **Int** ChildSubscriptionID – Identifier of a subscription that is subordinate to the specified subscription.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request -->
    <methodCall>
      <methodName>Execute</methodName>
        <params>
          <param>
            <value>
              <struct>
                <member>
                  <name>Server</name>
                  <value>BM</value>
                </member>
                <member>
                   <name>Method</name>
                   <value>ChildSubscriptionListGet_API</value>
                </member>
                <member>
                   <name>Params</name>
                   <value>
                    <array>
                      <data>

                        <!-- SubscriptionID. -->
                        <value><int>1000002</int></value>

                        <!-- SortNo. -->
                        <value><int>1</int></value>
                      </data>
                    </array>
                   </value>
                </member>
              </struct>
            </value>
          </param>
       </params>
     </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>

              <!-- ChildSubscriptionID1. -->
             <value><int>1000004</int></value>

              <!-- ChildSubscriptionID2. -->
             <value><int>1000005</int></value>
              <!-- ChildSubscriptionID3. -->
             <value><int>1000006</int></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

CountryListGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of configured countries for specified vendor.	2	10

Syntax

```
ListResult BM::CountryListGet_API(
  Int AccountID;
  Int SortNo.
)
returns
  Str(2) CountryID - two-letter country code;
  Str(30) Name - country name;
  Int FirstSortNo - country sort number in online store;
  Int TaxRegReq - always returns "0". The parameter is obsolete, left for
compatibility with PBA versions 4.2.x and earlier;
  Str Regular expression for zip code - regular expression for ZIP code
verification. Used when adding account from store, CP, and via API.
  Int TaxRegReqPers - signifies whether Tax Registration ID is required for
personal accounts: "0" - not required, "1" - optional, "2" - required;
  Int TaxRegReqComp - signifies whether Tax Registration ID is required for company
accounts: "0" - not required, "1" - optional, "2" - required.
  Str DefaultPhoneCountryCode - phone country code displayed by default in online
store after country is selected;
  Int NeedPhoneAreaCode - signifies whether to display Phone Area Code field in
online store: "0" - do not display field, "1" - display field;
  Int StateProvinceReq - signifies whether State/Province is required: "0" - not
required, "1" - required.
```

Special Notes

- **Int** AccountID - vendor account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Country ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>CountryListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Country ID -->
                 <value><string>ac</string></value>
                 <!-- Country Name -->
                 <value><string>Ascension Island</string></value>
                 <!-- Sort Number -->
                 <value><i4>-2</i4></value>
                 <!-- Obsolete Parameter -->
                 <value><i4>0</i4></value>
                 <!-- ZipCode RegExp -->
                 <value><string>^(.){0,10}$</string></value>
                 <!-- TaxRegReqPers -->
                 <value><i4>1</i4></value>
                <!-- TaxRegReqComp -->
                 <value><i4>0</i4></value>
                 <!-- DefaultPhoneCountryCode -->
                 <value><string>247</string></value>
                 <!-- NeedPhoneAreaCode -->
                 <value><i4>1</i4></value>
                 <!-- StateProvinceReq -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
       ...
              <value>
               <array>
                <data>
                 <!-- Country ID -->
                 <value><string>zw</string></value>
                 <!-- Country Name -->
                 <value><string>Zimbabwe</string></value>
                <!-- Sort Number -->
                 <value><i4>-2</i4></value>
                 <!-- Obsolete Parameter -->
                 <value><i4>0</i4></value>
                 <!-- ZipCode RegExp -->
                 <value>^(.){0,10}$</value>
                 <!-- TaxRegReqPers -->
                 <value><i4>0</i4></value>
                <!-- TaxRegReqComp -->
                 <value><i4>0</i4></value>
                 <!-- DefaultPhoneCountryCode -->
                 <value><string>63</string></value>
                <!-- NeedPhoneAreaCode -->
                 <value><i4>1</i4></value>
                 <!-- StateProvinceReq -->
                 <value><i4>0</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
```

```
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

CreateAccountAndPlaceOrder_API

	Parameters	
Description	**Passed**	**Returned**
Method is obsolete left for backward compatibility. Refer to the PlaceOrderAndAuthorize_API method (on page 391).	Depends on order structure	11 or 13

CreateManualOrder_API

Description	Parameters	
	Passed	**Returned**
Method places an order with the parameters specified.	Depends on order structure	1

Syntax

```
ItemResult BM::CreateManualOrder_API(
  Str SalesOrder: <attributes>
  Str OrdDet_1: <attributes>
  ...
  Str OrdDet_N: <attributes>
  Str TaxDet_1: <attributes>
  Str TaxDet_N: <attributes>
)
returns
  Int OrderID - ID of newly created order (do not mix with order number);
```

Special Notes:

Special notes are presented in the following subsections: the **Order Attributes** section (on page 172), the **Order Detail Attributes** section (on page 173) and the **Tax Detail Attributes** section (on page 175). Each section contains examples. Examples can be used to construct method calls for different purposes.

The whole method call examples are presented in the following subsections:

Example 1 - placing order with two order details, taxes are calculated by PBA;

Example 2 - placing order with one order detail, and two tax details to be applied;

The example of method response is presented in the **Example 1** subsection as it the same for both requests.

Order Attributes

Special Notes:

Str `SalesOrder: <attributes>` - defines the order general parameters. The `<attributes>` line comprises a set of pairs of the following format: `parameter=value`.

The following parameters are available:

- Int `AccountID` - the existing customer account ID to place the order for;

- **Str** `OrderTypeID` - the order type ID. The list of order types available can be viewed under the **System** > **Settings** > **Order Processing** > **Order Types** submenu of the Navigation tree. Each order type has its own order flow. It is recommended to place billing orders (BO) if there are no special requirements to order flow;
- **Str** `OrderStatusID` - status the order will be created in. The order processing will start from this status as well. The list of statuses available depends on order type. It can be viewed under the **Statuses** tab of the order type selected;
- **Str** `PromoCode` - a promotion code. The parameter is optional and can be skipped or passed empty;
- **Int** `BranchID` - ID of a sales branch to be assigned to the order. The parameter is optional and can be skipped or passed empty;
- **Int** `SalesID` - ID of a sales person to be assigned to the order. The parameter is optional and can be skipped or passed empty;
- **Int** `ConsolidationType` - signifies how the invoice is to be created for the order. Available options are: "0" - the invoice is generated according to the **Billing Type** parameter value set in the account customer class, "10" - the **Billing Type** parameter value set in the account customer class is ignored and invoice is generated as if the **Billing Type** parameter is set to the *Per Subscription* value, "20" - forced consolidation of the first order for **Billing Type** set to *Per Account*. The parameter is optional and can be skipped or passed empty;
- **Int** `SOPayToolID` - ID of the customer payment method (credit card or bank account) to be used to pay the order. The parameter is optional and can be skipped or passed empty. The customer default payment method will be used in this case;
- **Str** `FromIP` - customer IP address the order is placed from. The parameter is optional and can be skipped or passed empty;
- **Int** `OrderDate` - order creation date in Unix time format. The parameter is optional. If it is not specified, system current time is assigned;
- **Str** `Descr` - order general description, the order **Comments** field.

Example

```
<value>
      SalesOrder:
        AccountID=200;
        OrderTypeID=SO;
        OrderStatusID=NW;
        PromoCode=PromoCode5;
        BranchID=;
        SalesID=;
        ConsolidationType=10;
        SOPayToolID=;
        FromIP=10.20.30.40;
        OrderDate=;
        Descr =order description;
</value>
```

Order Detail Attributes

Special Notes

Str `OrdDet: <attributes>` - defines one order detail parameters. The number of the order details lines is not limited. The `<attributes>` line comprises a set of pairs of the following format: `parameter=value`.

The following parameters are available:

- **Dbl** `ExtendedPrice` - the detail line amount;
- **Dbl** `TaxAmt` - taxes total amount of the detail line;
- **Dbl** `TaxInclAmt` - taxes amount included in to the detail extended price;
- **Int** `TaxAlg` - signifies how taxes for this order detail are calculated. Available options are: "0" - taxes are to be calculated by PBA , "10" - PBA will set taxes specified in the passed tax details;
- **Dbl** `UnitPrice` - a unit price. Passed value is not used for any calculations. It is displayed in the **Unit Price** column of the order details list;
- **Int** `ServQty` - units quantity. Passed value is not used for any calculations. It is displayed in the **Quantity** column of the order details list;
- **Str** `ServUnitMeasure` - unit of measure. It is displayed in the **Unit of Measure** column of the order details list;
- **Int** `DurBillPeriodType` - billing period type. Available options are: "3" - year(s), "4" - month(s). It is displayed in the **Duration** column of the order details list;
- **Int** `DurBillPeriod` - biling period duration. It is displayed in the **Duration** column of the order details list. The parameter is optional and can be skipped, recommended value is "1".;
- **Int** `Duration` - duration. It is displayed in the **Duration** column of the order details list. The parameter is optional and can be skipped;
- **Int** `TaxCatID` - ID of an existing tax category applied to the order detail. It is displayed in the **Tax Category** column of the order details list;
- **Str** `Descr` - order detail description;

The following parameters are associated with the order detail for reporting purposes.

- **Int** `ResCatID` - ID of a resource category applied to the order detail. The parameter is optional and can be skipped;
- **Int** `planCategoryID` - ID of a plan category applied to the order detail. The parameter is optional and can be skipped;
- **Int** `subscriptionID` - ID of the customer existing subscription. The parameter is optional and can be skipped;
- **Int** `PlanPeriodID` - ID of a subscription period applied to the order detail. The parameter is optional and can be skipped;

- **Int** `PromoID` - ID of a promotion applied to the order detail. The parameter is optional and can be skipped;
- **Int** `DiscID` - ID of a discount applied to the order detail. The parameter is optional and can be skipped;
- **Int** `resourceID` - ID of a resource. The parameter is optional and can be skipped;

Example

```
<value>
      OrdDet:
        ExtendedPrice=20.3;
        TaxAmt=10.3;
        TaxInclAmt=0;
        DiscountAmt=0;
        TaxAlg=0;
        UnitPrice=1.0;
        ServQty=3.3;
        ServUnitMeasure=Unit;
        Duration=1.5;
        DurBillPeriod=1;
        DurBillPeriodType=4;
        TaxCatID=Default;
        ResCatID=;
        planCategoryID=;
        subscriptionID=;
        PlanPeriodID=;
        PromoID=;
        DiscID=;
        resourceID=;
        Descr=order detail description;
</value>
```

Tax Detail Attributes

Special Notes

Str `TaxDet: <attributes>` - defines one tax details to be applied to the order detail. The number of the tax details lines is not limited. Taxes specified here are applied to those order details that have `TaxAlg=10`. The `<attributes>` line comprises a set of pairs of the following format: `parameter=value`.

The following parameters are available:

- **Str** `TaxID` - ID of a tax that exists in PBA;
- **Dbl** `TxblAmt` - amount of the order that is subject for this tax;
- **Dbl** `TaxAmt` - tax amount applied to the order.

Example

```
<value>
      TaxDet:
        TaxID=VAT;
        TxblAmt=10.00;
        TaxAmt=10.3;
```

```
</value>
```

Order and Tax Details Consistency

The number of the order details and taxes in the method request is not limited. To make successful method request keep in mind the following rules:

- The tax category is the required parameter. Make sure that specified tax category ID exists in PBA.
- Taxes passed in tax details are applied to those order details that have the `TaxAlg=10`.
- Each tax detail and each order detail has the `TaxAmt` parameter. The `TaxAmt` values sum of all tax details should be equal to the `TaxAmt` values sum of all order details that have `TaxAlg=10`.
- If all order details have the `TaxAlg` set to `0` then the `TaxAmt` parameter of an order detail should be set to `0` and no tax details should be passed.

Example 1

The request places billing order (BO) with two order details, taxes are calculated by PBA.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>CreateManualOrder_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <value>
            SalesOrder:
                 AccountID=1000001;
                 OrderTypeID=BO;
                 OrderStatusID=NW;
                 PromoCode=windows_promotion;
                 BranchID=;
                 SalesID=;
                 ConsolidationType=10;
                 SOPayToolID=;
                 FromIP=10.20.30.40;
                 OrderDate=;
                 Descr =order placed via API;
           </value>
           <value>
            OrdDet:
                 ExtendedPrice=20.3;
                 TaxAmt=0;
                 TaxInclAmt=0;
                 DiscountAmt=0;
                 TaxAlg=0;
                 UnitPrice=1.0;
                 ServQty=3.3;
                 ServUnitMeasure=Unit;
                 Duration=1.5;
                 DurBillPeriod=1;
                 DurBillPeriodType=4;
                 TaxCatID=Default;
                 ResCatID=;
                 planCategoryID=;
                 subscriptionID=;
                 PlanPeriodID=;
                 PromoID=;
                 DiscID=;
                 resourceID=;
                 Descr=order detail 1 description;
           </value>
           <value>
            OrdDet:
                 ExtendedPrice=15;
                 TaxAmt=0;
                 TaxInclAmt=0;
                 DiscountAmt=0;
                 TaxAlg=0;
                 UnitPrice=5;
                 ServQty=3;
                 ServUnitMeasure=Unit;
                 Duration=1.5;
                 DurBillPeriod=1;
                 DurBillPeriodType=4;
                 TaxCatID=Default;
                 ResCatID=;
                 planCategoryID=;
                 subscriptionID=;
                 PlanPeriodID=;
                 PromoID=;
                 DiscID=;
                 resourceID=;
                 Descr=order detail 2 description;
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!--Placed order ID -->
              <value><i4>22</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

Example 2

The request places billing order (BO) with one order details and two tax details to be applied.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>CreateManualOrder_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <value>
            SalesOrder:
                 AccountID=1000001;
                 OrderTypeID=BO;
                 OrderStatusID=NW;
                 PromoCode=windows_promotion;
                 BranchID=;
                 SalesID=;
                 ConsolidationType=10;
                 SOPayToolID=;
                 FromIP=10.20.30.40;
                 OrderDate=;
                 Descr=order placed via API;
           </value>
           <value>
            OrdDet:
                 ExtendedPrice=60.3;
                 TaxAmt=20.3;
                 TaxInclAmt=0;
                 DiscountAmt=0;
                 TaxAlg=10;
                 UnitPrice=10.1;
                 ServQty=3;
                 ServUnitMeasure=Unit;
```

```
                Duration=1.5;
                DurBillPeriod=1;
                DurBillPeriodType=3;
                TaxCatID=Default;
                ResCatID=;
                planCategoryID=;
                subscriptionID=;
                PlanPeriodID=;
                PromoID=;
                DiscID=;
                resourceID=;
                Descr=order detail 1 description;
          </value>
          <value>
           TaxDet:
                TaxID=VAT;
                TxblAmt=60.30;
                TaxAmt=10.3;
          </value>
          <value>
           TaxDet:
                TaxID=DTB;
                TxblAmt=60.30;
                TaxAmt=10.0;
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

CreditCard2AccountAdd_API

	Parameters	
Description	**Passed**	**Returned**
Method adds credit card for specific account.	24	1

Syntax

```
ItemResult BM::CreditCard2AccountAdd_API(
  Int CustomerID;
  Str(40) CardHolderName;
  Str(20) CardNumber;
  Str(30) PaySystem;
  Str(5) ExpDate;
  Str(4) Cvv;
  Str(5) StartDate
  Str(4) IssueNumber
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) Country;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention;
  Int UseForAutoPayments.
)
returns
  Int CardID- payment method ID.
```

Special Notes

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the CardNumber argument must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42).

- **Int** CustomerID - customer account ID. Account must already exist in PBA;
- **Str(40)** CardHolderName - card holder name (as specified on the card);
- **Str(20)** CardNumber - credit card number (15 or 16 digits, no separators between them);
- **Str(30)** PaySystem - payment system name (Visa, Mastercard and the like). Payment system must be already set up in PBA;
- **Str(5)** ExpDate - credit card expiration date (MM/YY);
- **Str(4)** Cvv - credit card CVV code (3 or 4 digits depending on the card type);
- **Str(5)** StartDate - credit card start date. If it is not applicable, pass empty value;
- **Str(4)** IssueNumber - credit card issue number. It is required for some card types, for example, Solo and Switch. If it is not applicable, pass empty value;
- **Str(100)** Email - customer account e-mail address;
- **Str(80)** Address1 - customer account address (line 1/2);
- **Str(80)** Address2 - customer account address (line 2/2);
- **Str(40)** City - customer account address: city;
- **Str(80)** State - customer account address: state;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list at **System** > **Settings** > **Internationalization** > **Countries**;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list at **System** > **Settings** > **Internationalization** > **Countries**;
- **Str(4)** PhCountryCode - customer account contact information: phone country code.

- **Str(10)** PhAreaCode - customer account contact information: phone area code.
- **Str(20)** PhNumber - customer account contact information: phone number.
- **Str(10)** PhExtention - customer account contact information: phone extension.
- **Str(4)** FaxCountryCode - customer account contact information: fax country code.
- **Str(10)** FaxAreaCode - customer account contact information: fax area code.
- **Str(20)** FaxNumber - customer account contact information: fax number.
- **Str(10)** FaxExtention - customer account contact information: fax extension.
- **Int** UseForAutoPayments - defines whether new credit card should be used for automatic payments. Available options are:
    - 1 - mark new credit card as default payment method;
    - 2 - do NOT mark new credit card as default payment method.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>CreditCard2AccountAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OwnerAccountID -->
         <value><i4>1000001</i4></value>
         <!-- CardHolderName -->
         <value>JOHN SMITH</value>
         <!-- CardNumber -->
         <value>4111111111111111</value>
        <!-- PaySystem -->
         <value>Visa</value>
         <!-- ExpDate -->
         <value>05/10</value>
         <!-- Cvv -->
         <value>123</value>
         <!-- StartDate -->
         <value></value>
         <!-- IssueNumber -->
         <value></value>
         <!-- Email -->
         <value>jsmith@company.com</value>
         <!-- Address1 -->
         <value>Fifth Avenue</value>
         <!-- Address2 -->
         <value>Jersy</value>
         <!-- City -->
         <value>New York</value>
         <!-- State -->
         <value>NY</value>
        <!-- Zip -->
         <value>12354</value>
        <!-- CountryID -->
         <value>us</value>
         <!-- PhCountryCode -->
         <value>1</value>

         <!-- PhAreaCode -->
         <value>201</value>

        <!-- PhNumber -->
         <value>1235689</value>

         <!-- PhExtention -->
         <value>123</value>

         <!-- FaxCountryCode -->
         <value>1</value>

         <!-- FaxAreaCode -->
         <value>201</value>

         <!-- FaxNumber -->
         <value>1235689</value>

         <!-- FaxExtention -->
         <value>123</value>

         <!-- UseForAutoPayments -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added Credit Card ID -->
              <value><i4>24</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DeliverLinesForPeriodByCustomerListGet_API

	Parameters	
Description	**Passed**	**Returned**
This method returns the list of *deliver lines* that were created in a given period of time for a specified customer.	4	25

Syntax

```
BM::DeliverLinesForPeriodByCustomerListGet_API(
  Int FromDate;
  Int ToDate;
  Int CustomerID;
  Int SortNo.
)
returns
  Int Deliver Line ID;
  Int Detail ID;
  Int Resel Trans ID;
  Int Customer ID;
  Int SubscrID;
  Int Detail Type ID;
  Int PlanID;
  Int Resource ID;
  Int Parent Deliver Line ID;
  Int Deliver Line Status;
  DBL TaxAmt;
  DBL TaxInclAmt;
  DBL Extended Price;
  DBL TaxAmtPerMonth;
  DBL TaxInclAmtPerMonth;
  DBL Extended Price PerMonth;
  DBL Tax Percent;
  Int DL Start Date;
  Int DL End Date;
  Int DL Base Date;
  DBL PeriodInMonths;
  Str Unit Measure;
  DBL Resource Amount;
  Str Descr;
  Str SKU;
  DBL Discount Percent;
  DBL Unit Price;
  Str SalesID;
  Str BranchID;
  Int Billing Period Type;
  Int Billing Period.
```

Parameters Description

Input parameters:

- **Int** FromDate - The start date of the period for which the deliver lines are to be listed. The date is specified in the Unix Epoch format.
- **Int** ToDate - The end date of the period for which the deliver lines are to be listed. The date is specified in the Unix Epoch format.
- **Int** CustomerID - The identifier of the customer for whom the deliver lines created in the specified time period are to be listed.
- **Int** SortNo - This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, *N* is the number of a column):
  - N - The data is sorted by the N column in ascending order.
  - - N - The data is sorted by the N column in descending order.

Output parameters:

An array of the deliver lines data:

- **Int** DeliverLineID - Unique identifier of a deliver line.
- **Int** DetailID - Identifier of an order detail, from which the deliver line occurred in the system.

- **Int** ReselTransID - Identifier of a reseller transaction. This field is empty, in case the deliver line is not related to any reseller activity.
- **Int** CustomerID - Customer account identification.
- **Int** SubscrID - Identifier of a subscription in PBA
- **Int** DetailTypeID - Identifier of a detail type (plan setup, plan recurring, resource setup, etc.)
- **Int** PlanID - Identifier of the related service plan in PBA
- **Int** ResourceID - Resource identifier in PBA. The field is not empty only for additional resource charges.
- **Int** ParentDeliverLineID - Identifier of the parent deliver line. This field is not empty for both deliver line parts, which appear from an original deliver line after splitting.
- **Int** DeliverLineStatus - Status of the deliver line. This parameter can have one of the following values:
  - 1 – Active,
  - 999 – Deleted.
- **DBL** TaxAmt - Total tax amount applied to the deliver line.
- **DBL** TaxInclAmt - Included tax amount applied to the deliver line.
- **DBL** ExtendedPrice - Total price for the deliver line.
- **DBL** TaxAmtPerMonth - Total tax amount per month applied to the deliver line.
- **DBL** TaxInclAmtPerMonth - Included tax amount per month applied to the deliver line.
- **DBL** ExtendedPricePerMonth - Total price per month applied to the deliver line.
- **DBL** TaxPercent - Tax percent, applied to the deliver line.
- **Int** DLStartDate - Start date of the deliver line.
- **Int** DLEndDate - End date of the deliver line.
- **Int** DLBaseDate - Base date of the deliver line (usually taken from the start date of a subscription).
- **DBL** PeriodInMonths - Period between the start and end dates of the delivery line, in months.
- **Str** UnitMeasure - Unit of measure applicable to the order items. For example, “month(s)” is a unit of mesure for a subscription, and “GB” is a unit of mesure for additional disk space.
- **DBL** ResourceAmount - Resource amount for resource deliver lines. This field is empty for the subscription-related lines.
- **Str** Descr - Description of the deliver line.
- **Str** SKU - SKU of the deliver line.
- **DBL** Discount Percent - Amount of the discount applied to the deliver line.

- **DBL** Unit Price - Price of the purchased services specified per unit.
- **Str** SalesID - Identifier of the sales branch to which the purchase belongs.
- **Str** BranchID - Identifier of the sales person with which the purchase is associated.
- **Int** Billing Period Type - Type of the applicable billing period. The value of this parameter can be as follows:
  - 2 - Fixed Number of Months
  - 3 - Fixed Number of Years
  - 4 - Monthly on Statement Cycle Date
- **Int** Billing Period - Value of the Billing Period Type parameter.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>DeliverLinesForPeriodByCustomerListGet_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <!-- Int FromDate - A required parameter, the start date of the period for which the deliver lines are to be listed. The date is specified
in the Unix Epoch format.-->
           <value><i4>1354824000</i4></value>
           <!-- Int ToDate - A required parameter, the end date of the period for which the deliver lines are to be listed. The date is specified in
the Unix Epoch format.-->
           <value><i4>1357470000</i4></value>
           <!-- Int Customer ID - A required parameter, the identifier of a PBA customer account-->
           <value><i4>1000002</i4></value>
           <!-- Int SortNo - An optional parameter, the number of a column by which the data are to be sorted (enter 0 if you do not want the
data to be sorted). -->
           <value><i4>0</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
```

```
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <value><i4>17</i4></value>
                <value><i4>17</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1000002</i4></value>
                <value><i4>1000003</i4></value>
                <value><i4>100</i4></value>
                <value><i4>2</i4></value>
                <value><i4>0</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1</i4></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>40.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>40.000000</double></value>
                <value><double>0.000000</double></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><double>0.000000</double></value>
                <value><string></string></value>
                <value><double>0.000000</double></value>
                <value><string>Shared Hosting 2 Setup</string></value>
                <value><string></string></value>
                <value><double>1.000000/double></value>
                <value><double>0.000000</double></value>
                <value><string></string></value>
                <value><string></string></value>
                <value><i4>0</i4></value>
                <value><i4>0</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>21</i4></value>
                <value><i4>21</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1000002</i4></value>
                <value><i4>1000003</i4></value>
                <value><i4>120</i4></value>
                <value><i4>2</i4></value>
                <value><i4>100002</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1</i4></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>80.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>80.000000</double></value>
                <value><double>0.000000</double></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><double>0.000000</double></value>
                <value><string>item</string></value>
                <value><double>1.000000</double></value>
                <value><string>Boolean resource Setup</string></value>
                <value><string></string></value>
                <value><double>1.000000/double></value>
                <value><double>0.000000</double></value>
                <value><string></string></value>
                <value><string></string></value>
                <value><i4>0</i4></value>
                <value><i4>0</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>25</i4></value>
                <value><i4>19</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1000002</i4></value>
                <value><i4>1000003</i4></value>
                <value><i4>120</i4></value>
                <value><i4>2</i4></value>
                <value><i4>100001</i4></value>
                <value><i4>0</i4></value>
                <value><i4>1</i4></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>60.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>0.000000</double></value>
                <value><double>60.000000</double></value>
                <value><double>0.000000</double></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><i4>1354827600</i4></value>
                <value><double>0.000000</double></value>
                <value><string>item</string></value>
                <value><double>1.000000</double></value>
                <value><string>Discspace 1 Gb Setup</string></value>
                <value><string></string></value>
                <value><double>1.000000/double></value>
                <value><double>0.000000</double></value>
                <value><string></string></value>
                <value><string></string></value>
                <value><i4>0</i4></value>
                <value><i4>0</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
    </param>
  </params>
</methodResponse>
```

DeliverLinesForPeriodByVendorListGet_API

Description	Parameters	
	Passed	Returned
This method returns the list of *deliver lines* that were created in a given period of time for a specified vendor. **Note:** in the results returned, the pricing matches the Extra Precision configuration in PBA. For more details refer to the *Extra Precision in PBA Pricing* section in *PBA Provider's Guide.*	4	25

Syntax

```
BM::DeliverLinesForPeriodByVendorListGet_API(
  Int FromDate;
  Int ToDate;
  Int VendorID;
  Int SortNo.
)
returns
  Int Deliver Line ID;
  Int Detail ID;
  Int Resel Trans ID;
  Int Vendor ID;
  Int SubscrID;
  Int Detail Type ID;
  Int PlanID;
  Int Resource ID;
  Int Parent Deliver Line ID;
  Int Deliver Line Status;
  DBL TaxAmt;
  DBL TaxInclAmt;
  DBL Extended Price;
  DBL TaxAmtPerMonth;
  DBL TaxInclAmtPerMonth;
  DBL Extended Price PerMonth;
  DBL Tax Percent;
  Int DL Start Date;
  Int DL End Date;
  Int DL Base Date;
  DBL PeriodInMonths;
  Str Unit Measure;
  DBL Resource Amount;
  Str Descr;
  Str SKU;
```

```
DBL Discount Percent;
DBL Unit Price;
Str SalesID;
Str BranchID;
Int Billing Period Type;
Int Billing Period.
```

Parameters Description

Input parameters:

- **Int** FromDate - The start date of the period for which the deliver lines are to be listed. The date is specified in the Unix Epoch format.
- **Int** ToDate - The end date of the period for which the deliver lines are to be listed. The date is specified in the Unix Epoch format.
- **Int** VendorID - The identifier of the customer for whom the deliver lines created in the specified time period are to be listed.
- **Int** SortNo - This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, *N* is the number of a column):
  - N - The data is sorted by the N column in ascending order.
  - - N - The data is sorted by the N column in descending order.

Output parameters:

An array of the deliver lines data:

- **Int** DeliverLineID - Unique identifier of a deliver line.
- **Int** DetailID - Identifier of an order detail, from which the deliver line occurred in the system.
- **Int** ReselTransID - Identifier of a reseller transaction. This field is empty, in case the deliver line is not related to any reseller activity.
- **Int** CustomerID - Customer account identification.
- **Int** SubscrID - Identifier of a subscription in PBA
- **Int** DetailTypeID - Identifier of a detail type (plan setup, plan recurring, resource setup, etc.)
- **Int** PlanID - Identifier of the related service plan in PBA
- **Int** ResourceID - Resource identifier in PBA. The field is not empty only for additional resource charges.
- **Int** ParentDeliverLineID - Identifier of the parent deliver line. This field is not empty for both deliver line parts, which appear from an original deliver line after splitting.
- **Int** DeliverLineStatus - Status of the deliver line. This parameter can have one of the following values:
  - 1 – Active,

  - 999 – Deleted.
- **DBL** TaxAmt - Total tax amount applied to the deliver line.
- **DBL** TaxInclAmt - Included tax amount applied to the deliver line.
- **DBL** ExtendedPrice - Total price for the deliver line.
- **DBL** TaxAmtPerMonth - Total tax amount per month applied to the deliver line.
- **DBL** TaxInclAmtPerMonth - Included tax amount per month applied to the deliver line.
- **DBL** ExtendedPricePerMonth - Total price per month applied to the deliver line.
- **DBL** TaxPercent - Tax percent, applied to the deliver line.
- **Int** DLStartDate - Start date of the deliver line.
- **Int** DLEndDate - End date of the deliver line.
- **Int** DLBaseDate - Base date of the deliver line (usually taken from the start date of a subscription).
- **DBL** PeriodInMonths - Period between the start and end dates of the delivery line, in months.
- **Str** UnitMeasure - Unit of measure applicable to the order items. For example, “month(s)” is a unit of mesure for a subscription, and “GB” is a unit of mesure for additional disk space.
- **DBL** ResourceAmount - Resource amount for resource deliver lines. This field is empty for the subscription-related lines.
- **Str** Descr - Description of the deliver line.
- **Str** SKU - SKU of the deliver line.
- **DBL** Discount Percent - Amount of the discount applied to the deliver line.
- **DBL** Unit Price - Price of the purchased services specified per unit.
- **Str** SalesID - Identifier of the sales branch to which the purchase belongs.
- **Str** BranchID - Identifier of the sales person with which the purchase is associated.
- **Int** Billing Period Type - Type of the applicable billing period. The value of this parameter can be as follows:
  - 2 - Fixed Number of Months
  - 3 - Fixed Number of Years
  - 4 - Monthly on Statement Cycle Date
- **Int** Billing Period - Value of the Billing Period Type parameter.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>DeliverLinesForPeriodByVendorListGet_API</value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
            <!-- Int FromDate - A required parameter, the start date of the period for which the deliver lines are to be listed. The date is
specified in the Unix Epoch format.-->
            <value><i4>1354824000</i4></value>
            <!-- Int ToDate - A required parameter, the end date of the period for which the deliver lines are to be listed. The date is specified in
the Unix Epoch format.-->
            <value><i4>1360263600</i4></value>
            <!-- Int VendorID - A required parameter, the identifier of a PBA vendor or reseller account-->
            <value><i4>1000001</i4></value>
            <!-- Int SortNo - An optional parameter, the number of a column by which the data are to be sorted (enter 0 if you do not want the
data to be sorted). -->
            <value><i4>0</i4></value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodCall>
```

Response

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
```

```
         <value>
          <array>
           <data>
            <value>
             <array>
              <data>
               <value><i4>35</i4></value>
               <value><i4>35</i4></value>
               <value><i4>0</i4></value>
               <value><i4>1000001</i4></value>
               <value><i4>1000001</i4></value>
               <value><i4>110</i4></value>
               <value><i4>1</i4></value>
               <value><i4>0</i4></value>
               <value><i4>0</i4></value>
               <value><i4>1</i4></value>
               <value><double>0.000000</double></value>
               <value><double>0.000000</double></value>
               <value><double>69.600000</double></value>
               <value><double>0.000000</double></value>
               <value><double>0.000000</double></value>
               <value><double>80.000000</double></value>
               <value><double>0.000000</double></value>
               <value><i4>1357333200</i4></value>
               <value><i4>1359666000</i4></value>
               <value><i4>1354654800</i4></value>
               <value><double>0.870000</double></value>
               <value><string></string></value>
               <value><double>0.000000</double></value>
               <value><string>Resseler Plan 1 Recurring from 05-Jan-2013 through
31-Jan-2013. </string></value>
               <value><string></string></value>
               <value><double>1.000000/double></value>
               <value><double>0.000000</double></value>
               <value><string></string></value>
               <value><string></string></value>
               <value><i4>0</i4></value>
               <value><i4>0</i4></value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

DeliverLinesForPeriodByVendorMerge_API

	Parameters	
Description	**Passed**	**Returned**
This function merges the deliver lines for the specified vendor that were split within the specified time period.	3	1

Syntax

```
BM::DeliverLinesForPeriodByVendorMerge_API(
  Int FromDate;
  Int ToDate;
  Int VendorID;
)
returns
  Str Message.
```

Parameters Description

Input parameters:

- **Int** FromDate - The start date of the period within which the deliver lines were split. The date is specified in the Unix Epoch format
- **Int** ToDate - The end date of the period within which the deliver lines were split. The date is specified in the Unix Epoch format.
- **Int** VendorID - The identifier of the vendor for whom the deliver lines created in the specified time period are to merged.

Output parameter:

- **Str** Message - Message containing a text comment on the operation result.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all the comments from an actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>DeliverLinesForPeriodByVendorMerge_API</value>
      </member>
      <member>
       <name>Params</name>
       <value>
        <array>
         <data>
          <!-- Int FromDate - A required parameter, the start date of the period within which the split deliver lines are to be merged, specified in
the Unix Epoch format-->
          <value><i4>-1769385600</i4></value>
          <!-- Int ToDate - A required parameter, the end date of the period within which the split deliver lines are to be merged, specified in the
Unix Epoch format-->
          <value><i4>-1769385600</i4></value>
          <!-- Int VendorID - A required parameter, the identifier of a PBA vendor or reseller account-->
          <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <struct>
```

```
           <member>
            <name>Status</name>
            <value><string>Operation done.</string></value>
           </member>
          </struct>
         </value>
        </data>
       </array>
      </value>
     </member>
     <member>
      <name>TransactionID</name>
      <value><i4>33643</i4></value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

DeliverLinesForCustomerByDateSplit_API

	Parameters	
Description	**Passed**	**Returned**
The function splits the deliver lines for the specified customers by a given date.	2	1

Syntax

```
BM::DeliverLinesForCustomerByDateSplit_API(
  Int SplitDate;
  Int CustomerID;
)
returns
  Str Message.
```

Parameters Description

Input parameters:

- **Int** SplitDate - The split date, in the Unix Epoch format. As a result of the splitting, all the deliver lines, which were started before and finished after the given date, must be split into two parts.
- **Int** CustomerID - The identifier of the PBA customer account for which the deliver lines are to be split.

Output parameter:

- **Str** Message - Message containing a text comment on the operation result.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given onle for your convenience. PBA XMLRPC
does not parse xml comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>DeliverLinesForCustomerByDateSplit_API</value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
            <!-- Int ToDate - A required parameter, the date to split the deliver lines by specified in the Unix Epoch format-->
            <value><i4>1356984000</i4></value>
            <!-- Int CustomerID - A required parameter, the identifier of a PBA customer account-->
            <value><i4>1000003</i4></value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodCall>
```

Response

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <struct>
            <member>
             <name>Status</name>
             <value><string>Operation done.</string></value>
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DeliverLinesForVendorByDateSplit_API

Description	Parameters	
	Passed	**Returned**
The function splits the deliver lines for all the customers belonging to a given vendor by a given date.	2	1

Syntax

```
BM::DeliverLinesForVendorByDateSplit_API(
  Int SplitDate;
  Int VendorID;
)
returns
  Str Message.
```

Parameters Description

Input parameters:

- **Int** SplitDate - The split date, in the Unix Epoch format. As a result of the splitting, all the deliver lines, which were started before and finished after the given date must be split into two parts.
- **Int** VendorID - The identifier of the PBA vendor account for which the deliver lines are to be split.

Output parameter:

- **Str** Message - Message containing a text comment on the operation result.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given onle for your convenience. PBA XMLRPC
does not parse xml comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>DeliverLinesForVendorByDateSplit_API</value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
            <!-- Int ToDate - A required parameter, the date to split the deliver lines by specified in the Unix Epoch format-->
            <value><i4>1356984000</i4></value>
            <!-- Int VendorID - A required parameter, the identifier of a PBA vendor or reseller account-->
            <value><i4>1000001</i4></value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodCall>
```

Response

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <struct>
            <member>
             <name>Status</name>
             <value><string>Operation done.</string></value>
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DocumentsForPeriodByCustomerListGet_API

	Parameters	
Description	**Passed**	**Returned**
This method returns the list of details (identifiers and types) of the AR documents that were created in a given period of time for a specified customer.	4	3

Note: Only the documents having the *Open* or *Closed* status will be listed.

Syntax

```
ItemResult BM::DocumentsForPeriodByCustomerListGet_API(
  Int FromDate;
  Int ToDate;
  Int CustomerID;
  Int SortNo.
)
returns
  Int DocID;
  Int DocType;
  Int Status.
```

Parameters Description

Input parameters:

- **Int** FromDate - The start date of the period for which the AR document details are to be listed. The date is specified in the Unix Epoch format.
- **Int** ToDate - The end date of the period for which the AR document details are to be listed. The date is specified in the Unix Epoch format.
- **Int** CustomerID - The identifier of the customer for whom the details of the AR documents created in the specified time period are to be listed.
- **Int** SortNo - This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, *N* is the number of a column):
  - N - The data is sorted by the N column in ascending order.
  - - N - The data is sorted by the N column in descending order.

Output parameters:

An array of the AR document details data:

- **Int** DocID - An identifier of an AR document.
- **Int** DocType - A type of the AR document.
- **Int** Status - Status of the document. The status can be one of the following:
    - `1000` - Open
    - `2000` - Hold
    - `3000` - Closed
    - `4000` - Voided
    - `5000` - Refunded
    - `6000` - Cancelled
    - `7000` - Delayed

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse XML comments, so REMOVE all the comments from an actual request. -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>DocumentsForPeriodByCustomerListGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
          <!-- Int FromDate - A required parameter, the start date of the period for which the AR document details are to be listed. The date is
specified in the Unix Epoch format.-->
          <value><i4>1354046400</i4></value>
          <!-- Int ToDate - A required parameter, the end date of the period for which the AR document details are to be listed. The date is
specified in the Unix Epoch format.-->
          <value><i4>1359662400</i4></value>
          <!-- Int CustomerID - A required parameter, the identifier of a PBA account the AR documents data for which are to be displayed.-->
          <value><i4>1000003</i4></value>
          <!-- Int SortNo - An optional parameter, the number of a column by which the data are to be sorted (enter 0 if you do not want the data
to be sorted). ---->
          <value><i4>0</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <value><i4>1</i4></value>
                <value><i4>50</i4></value>
                <value><i4>1000</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>2</i4></value>
                <value><i4>20</i4></value>
                <value><i4>3000</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>11</i4></value>
                <value><i4>50</i4></value>
                <value><i4>4000</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>14</i4></value>
                <value><i4>20</i4></value>
                <value><i4>2000</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DocumentsForPeriodByVendorListGet_API

Description	Parameters	
	Passed	Returned
This method returns the list of details (identifiers and types) of the AR documents that were created in a given period of time for a specified vendor.	4	4

Note: Only the documents having the *Open* or *Closed* status will be listed.

Syntax

```
ItemResult BM::DocumentsForPeriodByVendorListGet_API(
  Int FromDate;
  Int ToDate;
  Int VendorID;
  Int SortNo.
)
returns
  Int DocID;
  Int DocType;
  Int AccountID;
  Int Status.
```

Parameters Description

Input parameters:

- **Int** FromDate - The start date of the period for which the AR document details are to be listed. The date is specified in the Unix Epoch format.
- **Int** ToDate - The end date of the period for which the AR document details are to be listed. The date is specified in the Unix Epoch format.
- **Int** VendorID - The identifier of the vendor for whom the details of the AR documents created in the specified time period are to be listed.
- **Int** SortNo - This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, *N* is the number of a column):
    - N - The data is sorted by the N column in ascending order.
    - - N - The data is sorted by the N column in descending order.

Output parameters:

An array of the following AR document details data:

- **Int** DocID - An identifier of an AR document.

- **Int** DocType - A type of the AR document.
- **Int** AccountID - The identifier of the customer account associated with the corresponding AR documents.
- **Int** Status - Status of the document. The status can be one of the following:
  - `1000` - Open
  - `2000` - Hold
  - `3000` - Closed
  - `4000` - Voided
  - `5000` - Refunded
  - `6000` - Cancelled
  - `7000` - Delayed

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse XML comments, REMOVE all the comments from an actual request. -->
  <methodCall>
   <methodName>Execute</methodName>
    <params>
     <param>
      <value>
       <struct>
        <member>
         <name>Server</name>
         <value>BM</value>
        </member>
        <member>
         <name>Method</name>
         <value>DocumentsForPeriodByVendorListGet_API</value>
        </member>
        <member>
         <name>Params</name>
         <value>
          <array>
           <data>
            <!-- Int FromDate - A required parameter, the start date of the period for which the AR document details are to be listed. The date is
specified in the Unix Epoch format.-->
            <value><i4>1262293200</i4></value>
            <!-- Int ToDate - A required parameter, the start date of the period for which the AR document details are to be listed. The date is
specified in the Unix Epoch format.-->
            <value><i4>1356984000</i4></value>
            <!-- Int CustomerID - A required parameter, the identifier of a PBA vendor or reseller account the AR documents data for which are
to be displayed.-->
            <value><i4>1000006</i4></value>
            <!-- Int SortNo - An optional parameter, the number of a column by which the data are to be sorted (enter 0 if you do not want the
data to be sorted). -->
            <value><i4>0</i4></value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <value><i4>5</i4></value>
                <value><i4>50</i4></value>
                <value><i4>1000005</i4></value>
                <value><i4>1000</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <value><i4>6</i4></value>
                <value><i4>20</i4></value>
                <value><i4>1000005</i4></value></data>
                <value><i4>1000</i4></value>
               </array>
              </value>
              <value>
               <array>
                <data>
                 <value><i4>34</i4></value>
                 <value><i4>50</i4></value>
                 <value><i4>1000009</i4></value>
                 <value><i4>3000</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

DomainExpirationDateGet_API

Description	Parameters	
	Passed	Returned
Method returns expiration date for specified domain.	2	5

Syntax

```
ItemResult DOMAINGATE::DomainExpirationDateGet_API(
  Int VendorID;
  Str(255) FullDomainName.
)
returns
  Int RegistrarID - ID of registrar plug-in domain is registered with.
  Int DomainID - domain ID.
  Int Expires - domain expiration date in Unix date format.
  Int RegisteredTo_AccountID - ID of customer account domain is registered for.
  Int PlanID - ID of service plan.
```

Special Notes

- **Int** VendorID - account ID of domain registration / transfer vendor (provider or reseller) [field is returned starting PBA release 4.3.2];
- **Str(255)** FullDomainName - domain name + TLD, without www prefix. For example, "domain.com".

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainExpirationDateGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorID -->
         <value><i4>1</i4></value>
         <!-- FullDomainName -->
         <value>domain.com</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Registrar ID -->
              <value><i4>1</i4></value>
              <!-- Domain ID -->
              <value><i4>3</i4></value>
              <!-- Expiration Date -->
              <value><i4>1275132977</i4></value>
              <!-- Domain owner Account ID -->
              <value><i4>1000001</i4></value>
              <!-- Service Plan ID -->
              <value><i4>2</i4></value>
             </data>
            </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodResponse>
```

DomainExtDataAdd_API

Description	Parameters	
	Passed	**Returned**
The method adds new extended data or updates existing one for *registered* domain. Extended data comprises specific parameters required by particular registrar. For example, to register a domain in the .net.au zone you are to specify your organization number - the ABN/ACN parameter. The list of parameters is stored in respective registrar plug-in container. Extended data provided for a particular domain can be viewed under the **Additional Information** tab of selected domain.	3 or more	1

Syntax

```
ErrorMessage DOMAINGATE::DomainExtDataAdd_API(
  Int DomainID;
  Str ParameterID_1;
  Str ParameterValue_1;
...
  Str ParameterID_N;
  Str ParameterValue_N;
)
returns
  Str ErrorMessage - message after the domain extended data has been added or
updated: "Domain ExtData has been added/updated successfully."
```

Special Notes

- **Int** DomainID - unique internal identifier of the registered domain name;
- **Str** ParameterID - ID of a parameter to be added or updated for specified domain. If you want to update value of existing parameter, take the parameter ID from the list under the **Additional Information** tab of selected domain. If you want to add new parameter, contact Parallels support to get the ID;
- **Str** ParameterValue - respective parameter value. Note, some parameters require specific values, contact Parallels support for details.

Example

Request

```
<?xml version="1.0"?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all  comments from actual request -->
<methodCall>
<methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
       <name>Server</name>
        <value>
         <string>DOMAINGATE</string>
        </value>
      </member>
      <member>
       <name>Method</name>
        <value>DomainExtDataAdd_API</value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <!-- DomainID -->
           <value>15</value>
           <!-- ParameterID -->
           <value>CA_IsATrademark</value>
           <!-- ParameterValue -->
          <value>1</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <struct>
            <member>
```

```
             <name>Status</name>
              <value><string>Domain ExtData has been added/updated
successfully</string>
            </value>
           </member>
          </struct>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

DomainInfoGet_API

Description	Parameters	
	Passed	Returned
Method returns information (contact and technical) for specific domain stored in PBA DB. Method does not return whois information. Result is returned for all domain subscriptions, including expired and terminated.	1	38

Syntax

```
ItemResult DOMAINGATE::DomainInfoGet_API(
  Str(255) FullDomainName.
)
returns
  Int AccountID - account ID of domain owner [field is returned starting PBA
release 4.3.2];
  Int VendorID -  account ID of domain registration / transfer vendor (provider or
reseller) [field is returned starting PBA release 4.3.2];
  Int DomainID - domain internal ID;
  Str(255) FullDomainName - domain name + domain zone;
  Str(40) DomainZoneID - domain zone;
  Str Status - domain internal status, possible values: "Registered", "Cancelled",
"Submitting for Renew", etc.;
  Str(40) External DNS - obsolete, always returns "External DNS";
  Int RegistrationPeriod - domain registration period (integer);
  Int startDate - domain subscription start date (Unix time);
  Int ExpirationDate - domain subscription expiration date (Unix time);
  Str(100) PrimaryNameServer - domain primary NS;
  Str(100) SecondaryNameServer - domain secondary NS;
  Str(100) ThirdNameServer - domain third NS;
  Str(100) FourthNameSever - domain fourth NS;
  Str(80) CompanyName - domain owner company name (for Company account type) or
first name + last name (for Personal account type);
  Str(30) FName - domain owner first name (personal contact);
  Str(30) MName - domain owner middle name (personal contact);
  Str(30) LName - domain owner last name (personal contact);
  Str(100) Email - domain owner e-mail address (personal contact);
  Str(40) getPhone - domain owner phone number (personal contact);
  Str(40) getFax - domain owner fax number (personal contact);
  Str(30) FName - domain owner first name (administrative contact);
  Str(30) MName - domain owner middle name (administrative contact);
  Str(30) LName - domain owner last name (administrative contact);
  Str(100) Email - domain owner e-mail address (administrative contact);
  Str(40) getPhone - domain owner phone number (administrative contact);
  Str(40) getFax - domain owner fax number (administrative contact);
  Str(30) FName - domain owner first name (billing contact);
  Str(30) MName - domain owner middle name (billing contact);
  Str(30) LName - domain owner last name (billing contact);
  Str(100) Email - domain owner e-mail address (billing contact);
  Str(40) getPhone - domain owner phone number (billing contact);
  Str(40) getFax - domain owner fax number (billing contact);
  Str(30) FName - domain owner first name (technical contact);
  Str(30) MName - domain owner middle name (technical contact);
  Str(30) LName - domain owner last name (technical contact);
  Str(100) Email - domain owner e-mail address (technical contact);
  Str(40) getPhone - domain owner phone number (technical contact);
  Str(40) getFax - domain owner fax number (technical contact);
```

Special Notes

Str(255) FullDomainName - domain name + TLD, without www prefix. For example, "domain.com".

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all  comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainInfoGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- FullDomainName -->
         <value>domain.com</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
```

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <!-- Account ID -->
            <value><i4>1000001</i4></value>
            <!-- Vendor ID -->
            <value><i4>1</i4></value>
            <!-- Domain ID -->
            <value><i4>3</i4></value>
            <!-- FullDomainName -->
            <value><string>domain.com</string></value>
            <!-- Domain zone ID -->
            <value><string>com</string></value>
            <!-- Status -->
            <value><string>Registered</string></value>
            <!-- Obsolete Parameter -->
            <value><string>external DNS</string></value>
            <!-- RegistrationPeriod -->
            <value><i4>1</i4></value>
            <!-- StartDate -->
            <value><i4>1243596978</i4></value>
            <!-- ExpirationDate -->
            <value><i4>1275132977</i4></value>
            <!-- PrimaryNameServer -->
            <value><string>1.1.1.1</string></value>
            <!-- SecondaryNameServer -->
            <value><string>2.2.2.2</string></value>
            <!-- ThirdNameServer -->
            <value><string></string></value>
            <!-- FourthNameServer -->
            <value><string></string></value>
            <!-- CompanyName -->
            <value><string>John M. Smith</string></value>
            <!-- PersnFName -->
            <value><string>John</string></value>
           <!-- PersnMName -->
            <value><string>M.</string></value>
           <!-- PersnLName -->
           <value><string>Smith</string></value>
          <!-- PersnEmail -->
           <value><string>jsmith@skywings.com</string></value>
           <!-- PersnPhNumber -->
           <value><string>1 703 8155670</string></value>
           <!-- PersnFaxNumber -->
           <value><string>1 703 8155670</string></value>
           <!-- AdminFName -->
           <value><string>John</string></value>
          <!-- AdminMName -->
           <value><string>M.</string></value>
           <!-- AdminLName -->
            <value><string>Smith</string></value>
            <!-- AdminEmail -->
            <value><string>jsmith@skywings.com</string></value>
           <!-- AdminPhNumber -->
            <value><string>1 703 8155670</string></value>
           <!-- AdminFaxNumber -->
            <value><string>1 703 8155670</string></value>
           <!-- BillFName -->
            <value><string>John</string></value>
            <!-- BillMName -->
            <value><string>M.</string></value>
            <!-- BillLName -->
            <value><string>Smith</string></value>
           <!-- BillEmail -->
            <value><string>jsmith@skywings.com</string></value>
           <!-- BillPhNumber -->
            <value><string>8155670</string></value>
           <!-- BillFaxNumber -->
            <value><string>1 703 8155670</string></value>
           <!-- TechFName -->
            <value><string>John</string></value>
          <!-- TechMName -->
            <value><string>M.</string></value>
           <!-- TechLName -->
            <value><string>Smith</string></value>
           <!-- TechEmail -->
            <value><string>jsmith@skywings.com</string></value>
           <!-- TechPhNumber -->
            <value><string>1 703 8155670</string></value>
           <!-- TechFaxNumber -->
           <value><string>1 703 8155670</string></value>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

DomainNamesAvailability_API

Description	Parameters	
	Passed	Returned
Method checks domain names availability with respective registrar.	4 or more	11

Syntax

```
ListResult DOMAINGATE::DomainNamesAvailability_API(
  Int OperationType;
  Int CountName;
  Str DomainName_1
  ...
  Str DomainName_N;
  Int CountAPlan;
  Int ActivePlanID_1
  ...
  Int ActivePlanID_N;
  Int CountIPlan;
  Int InactivePlanID_1
  ...
  Int InactivePlanID_N;
  Int CountTransferKey;
  Str TransferKey_1;
  ...
  Str TransferKey_N.
  )
returns
  Str FullDomainName - domain name + domain zone the way customer has input it;
  Str TechDomainName - domain name + domain zone the way it has been registered;
  Str FullDomainKernel - domain name the way customer has input it;
  Str TechDomainKernel - domain name the way it has been registered;
  Str FullDomainZone - domain zone the way customer has input it;
  Str TechDomainZone - domain zone the way it has been registered;
  Str PlanID - domain service plan ID;
  Str VendorID - vendor ID (provider or reseller);
  Str SuggestionLevel - indicates what domain name is registered: 0 - equally what
customer input, 1 - duplicated name, 2 - suggested available domain zones, 3 -
suggested domain kernel;
  Str  DomainStatus - indicates domain name check result: 0 - undefined check
result, 1 - unavailable domain name, 2 - available domain name;
  Str Message - message from respective registrar plug-in.
```

Special Notes

- **Int** OperationType - type of the operation to perform with a domain: 10 - register new domain, 20 - renew domain, 90 - transfer existing domain;
- **Int** CountName - number of domain names to be passed;
- **Str** DomainName - domain name with or without TLD;

- **Int** CountAPlan - number of service plans ID to be passed;
- **Int** ActivePlanID - ID of service plan for TLD customer selected to search domain name available in;
- **Int** CountIPlan - number of service plans ID additionally to be passed;
- **Int** InactivePlanID - ID of service plan for TLD to search available domain name in additionally;
- **Int** CountTransferKey - number of transfer keys to be passed;
- **Str** TransferKey - transfer key (if applicable).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
      </member>
       <member>
       <name>Method</name>
       <value>DomainNamesAvailability_API</value>
       </member>
       <member>
       <name>Params</name>
       <value>
        <array>
         <data>
        <!-- OperationType -->
         <value><i4>10</i4></value>
         <!-- CountName -->
         <value><i4>2</i4></value>
         <!-- DomainName  1 -->
         <value>mydomain</value>
         <!-- DomainName  2 -->
         <value>formedomain</value>
         <!-- CountAPlan -->
         <value><i4>1</i4></value>
        <!-- ActivePlanID -->
         <value><i4>1</i4></value>
         <!-- CountIPlan -->
         <value><i4>1</i4></value>
         <!-- InactivePlanID -->
         <value><i4>3</i4></value>
        <!-- CountTransferKey -->
         <value><i4>0</i4></value>
         <!-- TransferKey -->
         <value></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                <!-- Full Domain Name -->
                 <value><string>mydomain.biz</string></value>
                <!-- Tech Domain Name -->
                 <value><string>mydomain.biz</string></value>
                 <!-- Full Domain Kernel -->
                 <value><string>mydomain</string></value>
                <!-- Tech Domain Kernel -->
                 <value><string>mydomain</string></value>
                 <!-- Full Domain Zone -->
                 <value><string>biz</string></value>
                <!-- Tech Domain Zone -->
                 <value><string>biz</string></value>
                <!-- Plan ID -->
                 <value><string>1</string></value>
                 <!-- Vendor ID -->
                 <value><string>1</string></value>
                 <!-- SuggestionLevel -->
                 <value><string>0</string></value>
                 <!-- Domain Status -->
                 <value><string>1</string></value>
                 <!-- Message -->
                 <value><string>
                  Domain is not available for registration
                 </string></value>
                </data>
               </array>
              </value>
              <value>
               <array>
                <data>
                 <!-- Full Domain Name -->
                 <value><string>formedomain.biz</string></value>
                 <!-- Tech Domain Name -->
                 <value><string>formedomain.biz</string></value>
                 <!-- Full Domain Kernel -->
                 <value><string>formedomain</string></value>
                 <!-- Tech Domain Kernel -->
                 <value><string>formedomain</string></value>
                <!-- Full Domain Zone -->
                 <value><string>biz</string></value>
```

```
                  <!-- Tech Domain Zone -->
                  <value><string>biz</string></value>

                  <!-- Plan ID -->
                  <value><string>1</string></value>

                 <!-- Vendor ID -->
                  <value><string>1</string></value>

                 <!-- SuggestionLevel -->
                  <value><string>0</string></value>

                 <!-- Domain Status -->
                  <value><string>2</string></value>

                 <!-- Message -->
                  <value><string>
                   Domain available for registration
                  </string></value>
                 </data>
                </array>
               </value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
   </params>
 </methodResponse>
```

DomainPlanGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns specified domain service plan details.	1	15

Syntax

```
ItemResult DOMAINGATE::DomainPlanGet_API(
  Int PlanID.
)
returns
  Int PlanID - service plan ID;
  Str(60) Name - service plan name;
  Str(1024) ShortDescription - service plan short description;
  Str(1024) LongDescription - service plan long description;
  Int CategoryID - service plan sales category ID;
  Str STType - service plan provisioning gate, for example "DOMAINGATE";
  Int IsPublished - signifies if service plan is published or not, disabled by its
service template or by provider:
  "0" - published,
  "1" - not published,
  "2" - disabled by Template,
  "3" - not published and disabled by template;
  "4" - disabled by provider;
  "5" - not published and disabled by provider;
  "6" - disabled by provider and by template;
  "7" - not published and disabled by provider and by template;
  Int GroupID - ID of unique group service plan is included in;
  Int RecurringType - service plan recurring fee collection method (Charge for
Subscription parameter): "10" - before billing period; "20" - after billing period;
"30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - service plan billing period type: "2" - billing period is
a fixed number of months; "3" - billing period is a fixed number of years; "4" -
subscription is billed monthly on statement cycle date;
  Int BillingPeriod - service plan billing period;
  Str DomainZoneID - service plan domain zone ID;
  Int ShowPriority - order number of service plan in online store;
  Int IsInternalChecker - signifies whether Dummy Registrar ("1") or one of
external registrars ("0") is used as registrar for domain zone;
  Int Selected by Default - signifies whether domain zone is selected by default
("1") when customer registers domain is store / CP. Configured under domain
service template properties.
```

Special Notes

Int PlanID - ID of service plan.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainPlanGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>11</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Service Plan ID -->
               <value><i4>11</i4></value>
              <!-- Service Plan Name -->
               <value><string>.org.uk</string></value>
               <!-- Short Description -->
               <value><string>Get a domain!</string></value>
               <!-- Long Description -->
               <value><string>Get a domain!</string></value>
              <!-- Sales Category ID -->
               <value><i4>12003</i4></value>
               <!-- Service Gate -->
               <value><string>DOMAINGATE</string></value>
               <!-- IsPublished -->
               <value><i4>0</i4></value>
               <!-- Unique GroupID -->
               <value><i4>2034</i4></value>
               <!-- RecurringType -->
               <value><i4>10</i4></value>
               <!-- Billing Period Type -->
               <value><i4>4</i4></value>
              <!-- Billing Period -->
               <value><i4>1</i4></value>
               <!-- DomainZoneID -->
               <value><string>org.uk</string></value>
               <!-- ShowPriority -->
               <value><i4>-2</i4></value>
               <!-- IsInternalChecker -->
               <value><i4>1</i4></value>
              <!-- Selected by Default -->
               <value><i4>0</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

DomainPlanListAvailableGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of published domain service plans included in specified sales category.	3	15

Syntax

```
ListResult DOMAINGATE::DomainPlanListAvailableGet_API(
  Int AccountID;
  Int CategoryID;
  Int SortNo.
)
returns
  Int PlanID - service plan ID;
  Str(60) name - service plan name;
  Str(1024) shortDescription - service plan short description;
  Str(1024) longDescription - service plan long description;
  Int CategoryID - service plan sales category ID;
  Str STType - service plan provisioning gate, for example "DOMAINGATE";
  Int IsPublished - signifies if service plan is published: "0" - published, "1" -
not published. Method returns only published plan, so the parameter is always
"0";
  Int GroupID - ID of unique group service plan is included in;
  Int RecurringType - service plan recurring fee collection method (Charge for
Subscription parameter): "10" - before billing period; "20" - after billing period;
"30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - service plan billing period type: "2" - billing period is
a fixed number of months; "3" - billing period is a fixed number of years; "4" -
subscription is billed monthly on statement cycle date;
  Int BillingPeriod - service plan billing period;
  Str DomainZoneID - service plan domain zone ID;
  Int ShowPriority - order number of service plan in online store;
  Int IsInternalChecker - signifies whether Dummy Registrar ("1") or one of
external registrars ("0") is used as registrar for domain zone;
  Int Selected by Default - signifies whether domain zone is selected by default
("1") when customer registers domain is store / CP. Configured under domain
service template properties.
```

Special Notes

- **Int** AccountID - account ID of sales category owner (vendor);
- **Int** CategoryID - domain sales category ID;
- **Int** SortNo - defines how output data is sorted: "1" - the output is sorted by plan ID in ascending order, "2" - the output is sorted by plan name in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainPlanListAvailableGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1</i4></value>
         <!-- CategoryID -->
         <value><i4>2</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
```

```
             <data>
              <!-- Service Plan ID -->
              <value><i4>11</i4></value>
              <!-- Service Plan Name -->
              <value><string>.com</string></value>
              <!-- Short Description -->
              <value><string>Get a domain!</string></value>
              <!-- Long Description -->
              <value><string>Get a domain!</string></value>
              <!-- Sales Category ID -->
              <value><i4>2</i4></value>
              <!-- Service Gate -->
              <value><string>DOMAINGATE</string></value>
             <!-- IsPublished -->
              <value><i4>0</i4></value>
              <!-- Unique GroupID -->
              <value><i4>204</i4></value>
              <!-- RecurringType -->
              <value><i4>10</i4></value>
              <!-- Billing Period Type -->
              <value><i4>4</i4></value>
              <!-- Billing Period -->
              <value><i4>1</i4></value>
              <!-- DomainZoneID -->
              <value><string>.com</string></value>
              <!-- ShowPriority -->
              <value><i4>3</i4></value>
              <!-- IsInternalChecker -->
              <value><i4>1</i4></value>
              <!-- Selected by Default -->
              <value><i4>0</i4></value>
             </data>
            </array>
           </value>
...
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

DomainPlanPeriodListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of subscription periods of specified domain service plan.	2	14

Syntax

```
ListResult DOMAINGATE::DomainPlanPeriodListGet_API(
  Int PlanID;
  Int SortNo.
)
returns
  Int PlanPeriodID - service plan subscription period ID;
  Int Period - subscription period duration (for example, 1 if subscription period
is 1 year);
  Int PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 -
Month(s), 3 - Year(s);
  Int Trial - whether subscription period is trial (free): 0 - No, 1 - Yes;
  Double SetupFee - subscription setup fee. It is initial and one time charge
which is taken for subscription for this plan;
  Double SubscriptionFee - subscription recurring fee. It is taken once a billing
period;
  Double RenewalFee - subscription renewal fee. It is charged for subscription
renewal;
  Double TransferFee - subscription transfer fee. It is charged for domain
transfer and relevant for domain plans only;
  Double NonRefundableAmt - subscription non-refundable amount. The sum is not
refunded to customer when refund is claimed (on subscription cancellation, for
example);
  Int RefundPeriod - subscription refund period duration in days (for example, 5
if refund period is 5 days). During refund period customer can cancel his
subscription and get all money back, except non-refundable amount;
  Int Enabled - whether subscription period is active : 0 - No, 1 - Yes;
  Int NumberOfPeriods - number of billing periods within subscription period. This
parameter is calculated automatically and depends on Billing Period Type parameter of
service plan. For example, if Billing Period Type is "Fixed Number of Months" and
subscription period is 2 years long, there will be 24 billing periods, meaning
customer will be charged recurring fee 24 times;
  Str(1024) FeeText - text displayed as description of period fees, for example
"$9.99 setup + $18/month";
  Int SortNumber - order in which subscription periods are shown in online store
and CP: "1" - on top, "2" - below it, etc.
```

Special Notes

- **Int** PlanID - domain service plan ID;

- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (PlanPeriod ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainPlanPeriodListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>1</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Plan period ID -->
                 <value><i4>5</i4></value>
                 <!-- Period duration -->
                 <value><i4>1</i4></value>
                 <!-- Period type -->
                 <value><i4>3</i4></value>
                 <!-- Trial -->
                 <value><i4>0</i4></value>
                 <!-- Setup Fee -->
                 <value><double>1.000000</double></value>
                 <!-- Subscription Fee -->
                 <value><double>2.000000</double></value>
                 <!-- Renewal Fee -->
                 <value><double>0.000000</double></value>
                 <!-- Transfer Fee -->
                 <value><double>0.000000</double></value>
                 <!-- NonRefundable Amount -->
                 <value><double>0.000000</double></value>
                 <!-- Refund period -->
                 <value><i4>0</i4></value>
                <!-- Enabled -->
                 <value><i4>1</i4></value>
                 <!-- Number of periods -->
                 <value><i4>12</i4></value>
              <!-- Fee text -->
                 <value><string> </string></value>
                <!-- Sort number -->
                <value><i4>1</i4></value>
               </data>
              </array>
            </value>
             <value>
              <array>
               <data>
                <!-- Plan period ID -->
                <value><i4>6</i4></value>
                <!-- Period duration -->
                <value><i4>2</i4></value>
                <!-- Period type -->
                <value><i4>3</i4></value>
```

```
                              <!-- Trial -->
                              <value><i4>0</i4></value>

                              <!-- Setup Fee -->
                              <value><double>2.000000</double></value>

                              <!-- Subscription Fee -->
                              <value><double>5.000000</double></value>

                              <!-- Renewal Fee -->
                              <value><double>0.000000</double></value>

                              <!-- Transfer Fee -->
                              <value><double>0.000000</double></value>

                              <!-- NonRefundable Amount -->
                              <value><double>0.000000</double></value>

                              <!-- Refund period -->
                              <value><i4>0</i4></value>

                              <!-- Enabled -->
                              <value><i4>1</i4></value>

                              <!-- Number of periods -->
                              <value><i4>24</i4></value>

                              <!-- Fee text -->
                              <value><string> </string></value>

                              <!-- Sort number -->
                              <value><i4>2</i4></value>
                             </data>
                            </array>
                           </value>
                          </data>
                         </array>
                        </value>
                       </data>
                      </array>
                     </value>
                    </member>
                   </struct>
                  </value>
                 </param>
                </params>
               </methodResponse>
```

DomainSubscrAdd_API

Description	Parameters	
	Passed	**Returned**
Method creates domain subscription for a domain. Subscription information is added to PBA database and domain specific information is added to DOMAINGATE table.	30	2

If the passed domain name is already presented in the PBA database except the *Cancelled* status, an exception is returned.

Note: since version 5.4 the values of Login and Password are not passed to PBA.

Syntax

```
ItemResult DOMAINGATE::DomainSubscrAdd_API(
  Int ParentID;
  Int Status;
  Int ServStatus;
  Int StartDate;
  Int ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int SubscriptionPeriod;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate;
  Int RegistrarID;
  Str(255) FullDomainName;
  Str(40) Login;
  Str(32) Password;
  Str(40) PrimaryNameServer;
  Str(40) SecondaryNameServer;
  Str(40) ThirdNameServer;
  Str(40) FourthNameServer;
  Int OwnerID;
  Int AdminID;
  Int BillingID;
  Int TechID.
)
returns
  Int SubscriptionID - ID of the created subscription;
  Int DomainID - ID of the domain.
```

Special Notes

- **Int** ParentID - ID of the parent hosting subscription;
- **Int** Status - subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40 - "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 - "Administrative Hold", 85 - "Credit Hold", 89 - "Credit and Administrative Hold");
- **Int** ServStatus - subscription service status (10 - ''Not Provisioned'', 20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Int** StartDate - the date of subscription start in Unix date format;
- **Int** ExpirationDate - expiration date of the subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;

- **Int** PlanID - ID of domain subscription service plan;
- **Int** SubscriptionPeriod - subscription period duration (for example, 1 if subscription period is 1 year);

> **Note:** subscription period type (Days, Months, Years) is not passed with the method as subscription period type for domain registration is always "Years".

- **Double** SetupFee - subscription setup fee;
- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

> **Note:** fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days (for example, 5 if refund period is 5 days);
- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);
- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format;
- **Int** RegistrarID - ID of registrar plug-in domain is registered with. The list of supported registrar plug-ins can be accessed from the **System** > **Settings** > **Domains** submenu of the Navigation tree;
- **Str(255)** FullDomainName - domain name + TLD, without www prefix. For example, "domain.com";
- **Str(40)** Login - login to registrar;
- **Str(32)** Password - password to registrar;
- **Str(40)** PrimaryNameServer - domain primary NS;
- **Str(40)** SecondaryNameServer - domain secondary NS;
- **Str(40)** ThirdNameServer - domain third NS;
- **Str(40)** FourthNameServer - domain fourth NS;

> **Note:** primary and secondary name servers addresses are mandatory, the rest of name servers are optional.

- **Int** OwnerID - customer account user ID. The user contact information is used as owner contact by registrar;
- **Int** AdminID - customer account user ID. The user contact information is used as administrative contact by registrar;

- **Int** BillingID - customer account user ID. The user contact information is used as billing contact by registrar;
- **Int** TechID customer account user ID The user contact information is used as technical contact by registrar.

Note: all four contacts can be of the same user or of four different users.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
       </member>
       <member>
        <name>Method</name>
        <value>DomainSubscrAdd_API</value>
        </member>
        <member>
        <name>Params</name>
        <value>
         <array>
         <data>
         <!-- Parent  subscriptionID -->
         <value><i4>100056</i4></value>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
         <!-- startDate -->
         <value><i4>1234586</i4></value>
         <!-- ExpirationDate -->
         <value><i4>1234656</i4></value>
         <!-- AccountID -->
         <value><i4>1234545</i4></value>
         <!-- PlanID -->
         <value><i4>1243</i4></value>
         <!-- Period -->
         <value><i4></i4></value>
         <!-- SetupFee -->
         <value><double>13</double></value>
        <!-- RecurringFee -->
         <value><double>16</double></value>
         <!-- RenewalFee -->
         <value><double>0</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
         <!-- NonRefundableAmt -->
         <value><double>13</double></value>
         <!-- RefundPeriod -->
         <value><i4>5</i4></value>
         <!-- BillingPeriodType -->
         <value><i4>3</i4></value>
         <!-- BillingPeriod -->
         <value><i4>1</i4></value>
         <!-- LastBillDate -->
         <value><i4>123455</i4></value>
         <!-- NextBillDate -->
         <value><i4>123454</i4></value>
         <!-- RegistrarID -->
         <value><i4>8</i4></value>
         <!-- FullDomainName -->
         <value>domain.com</value>
         <!-- Login -->
         <value>login</value>
         <!-- Password -->
         <value>qwerty</value>
         <!-- PrimaryNameServer -->
         <value>ns1.provider.com</value>
         <!-- SecondaryNameServer -->
         <value>ns2.provider.com</value>
         <!-- ThirdNameServer -->
         <value>ns3.provider.com</value>
         <!-- FourthNameServer -->
         <value>ns4.provider.com</value>
         <!-- OwnerUsersID -->
         <value><i4>101</i4></value>
         <!-- AdminUsersID -->
         <value><i4>1000062</i4></value>
         <!-- BillingUsersID -->
         <value><i4>1000061</i4></value>
         <!-- TechUsersID -->
         <value><i4>1000063</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added subscription ID -->
              <value><i4>1000038</i4></value>
             <!-- Added domain ID -->
              <value><i4>18</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DomainSubscrWithIDAdd_API

	Parameters	
Description	**Passed**	**Returned**
Method creates domain subscription for *registered* domain with explicitly defined subscription ID. Subscription information is added to PBA database and domain specific information is added to DOMAINGATE table.	30	1

Syntax

```
ItemResult DOMAINGATE::DomainSubscrWithIDAdd_API(
  Int SubscriptionID;
  Int ParentID;
  Int Status;
  Int ServStatus;
  Str startDate;
  Str ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int SubscriptionPeriod;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate;
  Int RegistrarID;
  Str(255) FullDomainName;
  Str(40) Login;
  Str(32) Password;
  Str(40) PrimaryNameServer;
  Str(40) SecondaryNameServer;
  Str(40) ThirdNameServer;
  Str(40) FourthNameServer;
  Int OwnerUsersID;
  Int AdminUsersID;
  Int BillingUsersID;
  Int TechUsersID.
)
returns
  Int DomainID - unique domain identifier.
```

Special Notes

- **Int** SubscriptionID - new subscription ID;
- **Int** ParentID - ID of the parent subscription (hosting);
- **Int** Status - subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40 - "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 - "Administrative Hold", 85 - "Credit Hold", 89 - "Credit and Administrative Hold");
- **Int** ServStatus - subscription service status (10 - ''Not Provisioned'', 20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Str** startDate - the date of the subscription start in Unix date format;
- **Str** ExpirationDate - expiration date of the subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;
- **Int** PlanID - ID of domain subscription service plan;
- **Int** SubscriptionPeriod - subscription period duration (for example, 1 if subscription period is 1 year);

  Note: subscription period type (Days, Months, Years) is not passed with the method as subscription period type for domain registration is always "Years".

- **Double** SetupFee - subscription setup fee;

- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

Note: fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days (for example, 5 if refund period is 5 days);
- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);
- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format;
- **Int** RegistrarID - ID of registrar plug-in domain is registered with. The list of supported registrar plug-ins can be accessed from the **External Systems Director** > **Domain Manager** > **Registrars** submenu of the Navigation tree;
- **Str(255)** FullDomainName - domain name + TLD, without www prefix. For example, "domain.com";
- **Str(40)** Login - login to registrar;
- **Str(32)** Password - password to registrar;
- **Str(40)** PrimaryNameServer - domain primary NS;
- **Str(40)** SecondaryNameServer - domain secondary NS;
- **Str(40)** ThirdNameServer - domain third NS;
- **Str(40)** FourthNameServer - domain fourth NS;

Note: primary and secondary name servers addresses are mandatory, the rest of name servers are optional.

- **Int** OwnerUsersID - customer account user ID. The user contact information is used as owner contact by registrar;
- **Int** AdminUsersID - customer account user ID. The user contact information is used as administrative contact by registrar;
- **Int** BillingUsersID - customer account user ID. The user contact information is used as billing contact by registrar;
- **Int** TechUsersID - customer account user ID The user contact information is used as technical contact by registrar.

Note: all four contacts can be of the same user or of four different users.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>DOMAINGATE</value>
      </member>
       <member>
        <name>Method</name>
        <value>DomainSubscrWithIDAdd_API</value>
       </member>
       <member>
       <name>Params</name>
       <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>100023</i4></value>
         <!-- Parent  subscriptionID -->
         <value><i4>120003</i4></value>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
         <!-- startDate -->
         <value><i4>14588765623</i4></value>
         <!-- ExpirationDate -->
         <value><i4>12154566563</i4></value>
         <!-- AccountID -->
         <value><i4>1000028</i4></value>
         <!-- PlanID -->
         <value><i4>2035</i4></value>
         <!-- Period -->
         <value><i4>1203</i4></value>
         <!-- SetupFee -->
         <value><double>13</double></value>
         <!-- SubscriptionFee -->
         <value><double>10</double></value>
         <!-- RenewalFee -->
         <value><double>0</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
         <!-- NonRefundableAmt -->
         <value><double>10</double></value>
         <!-- RefundPeriod -->
         <value><i4>5</i4></value>
         <!-- BillingPeriodType -->
         <value><i4>2</i4></value>
         <!-- BillingPeriod -->
         <value><i4>1</i4></value>
         <!-- LastBillDate -->
         <value><i4>121111113</i4></value>
         <!-- NextBillDate -->
         <value><i4>14545454523</i4></value>
         <!-- RegistrarID -->
         <value><i4>103</i4></value>
         <!-- FullDomainName -->
         <value>domain.com</value>
         <!-- Login -->
         <value>login</value>
         <!-- Password -->
         <value>qwerty</value>
         <!-- PrimaryNameServer -->
         <value>ns1.provider.com</value>
         <!-- SecondaryNameServer -->
         <value>ns2.provider.com</value>
         <!-- ThirdNameServer -->
         <value>ns3.provider.com</value>
         <!-- FourthNameServer -->
         <value>ns4.provider.com</value>
         <!-- OwnerUsersID -->
         <value><i4>101</i4></value>
         <!-- AdminUsersID -->
         <value><i4>102</i4></value>
         <!-- BillingUsersID -->
         <value><i4>101</i4></value>
         <!-- TechUsersID -->
         <value><i4>103</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added domain ID -->
```

```
              <value><i4>15</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

DomainTransferKeyIsRequired_API

Description	Parameters	
	Passed	Returned
Method verifies if transfer key is required.	1	1

Syntax

```
ItemResult DOMAINGATE::DomainTransferKeyIsRequired_API(
  Int PlanID.
)
returns
  Int TransferKeyRequired - whether Transfer Key is required (0 - No, 1 - Yes).
```

Special Notes

Int PlanID - domain service plan ID.

Example

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainTransferKeyIsRequired_API</value>
     </member>
     <member>
```

```
      <name>Params</name>
      <value>
       <array>
        <data>
        <!-- PlanID -->
         <value><i4>1045</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <!-- TransferKeyRequired=0 - not required -->
               <i4><string>0</string></i4>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
   </param>
  </params>
 /methodResponse>
```

DomainValidate_API

Description	Parameters	
	Passed	**Returned**

Method gets domain name and domain plan ID and returns the list of domain availability check result. **Note:** the method is obsolete and should not be used.	2	4

Syntax

```
ListResult DOMAINGATE::DomainValidate_API(
  Str(40) DomainName;
  Int PlanID.
  Int SortNo
 )
returns
  Str(40) DomainName - domain name without TLD;
  Int DomainZone - TLD;
  Int IsSuggested - whether domain name is suggested by registrar (0 - No, 1 -
Yes);
  Int IsAvailable - whether domain is available(0 - No, 1 - Yes).
```

Special Notes

- **Str(40)** DomainName - domain name + TLD, without www prefix. For example, "domain.com";
- **Int** PlanID - domain service plan ID;
- **Int** SortNo - defines how output data is sorted: "1" - the output is sorted by plan ID in ascending order, "2" - the output is sorted by plan name in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>DomainValidate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- DomainName -->
         <value>domain</value>
        <!-- PlanID -->
         <value><i4>1</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
```

```
                <data>
                 <!-- Domain Name -->
                 <value><string>domain</string></value>
                <!-- Domain zone -->
                 <value><string>com</string></value>
                 <!-- IsSuggested -->
                 <value><i4>0</i4></value>
                <!-- IsAvailable -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
  </params>
 </methodResponse>
```

EulaListForOrderGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of EULA applicable to specified sales order.	2	1

Syntax

```
ListResult BM::EulaListForOrderGet_API(
  Int OrderID;
  Int SortNo.
)
returns
  Str(16384) EulaURL - URL of the EULA page.
```

Special Notes

- **Int** OrderID - sales order ID;
- **Int** SortNo - defines how output data is sorted. The method returns one column list, so "1" - the output is sorted by EULA in ascending order. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>EulaListForOrderGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OrderID -->
         <value><i4>112</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <!-- Eula URL -->
                <value><string>
                 I have read and I accept the
                 &lt;a href=&quot;example.com/customer-agreement.html
                 &quot;&gt;Provider, Inc. Terms of Service&lt;/a&gt;.
                 I also understand that Domain Name registration orders
                 cannot assure availability and are subject to these
                 same Terms and Conditions.
                </string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

EnablingResourceByPlanListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of enabling resources for all child resource categories related to the specified service plan.	3	5

Syntax

```
ListResult BM::EnablingResourceByPlanListGet_API(
  Int PlanID;
  Int SortNo;
)
returns
  Int ResCatID - child resource category ID, enabled by the following resource;
  Int ResourceID - enabling resource ID;
  Int PlanRateID - resource rate ID.
```

Special Notes

- **Int** PlanID - service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 2 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (ResCat ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>EnablingResourceByPlanListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>18</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Resource Category ID -->
                 <value><i4>10110</i4></value>
                 <!-- Resource ID -->
                 <value><i4>10005</i4></value>
                 <!-- Resource rate ID -->
                 <value><i4>36</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

GeneratePassword_API

	Parameters	
Description	**Passed**	**Returned**
The method returns generated password.	0	1

Syntax

```
ItemResult BM::GeneratePassword_API()
returns
  Str Password.
```

Parameters Description

- **Str** Password – generated password which complies with the requirements of the "Strong" password quality level.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GeneratePassword_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <name>Status</name>
               <value><string>
                    Hero8unruly!Eerily
             </string></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetAccountListByCustomAttribute_API

Description	Parameters	
	Passed	**Returned**
The method returns the list of accounts which contain specified custom attribute.	3	2

Syntax

```
ListResult BM::GetAccountListByCustomAttribute_API(
  Str AttributeID;
  Str AttributeValue;
  Int SortNo.
)
returns
  Int AccountID;
  Str CompanyName.
```

Parameters Description

Input Parameters:

- **Str** AttributeID – Identifier of a custom attribute that is set as the search criteria.
- **Str** AttributeValue – Value of the attribute that is set as the search criteria.
- **Int** SortNo – Parameter defining how the output data is sorted. The parameter is set the following way:
  - 1 – The data is sorted by the first column in ascending order.
  - 2 – The data is sorted by the second column in ascending order.
  - -1 – The data is sorted by the first column in descending order.
  - -2 – The data is sorted by the second column in descending order.

Output Parameters:

An array of items that contain the following attribute data:

- **Int** AccountID – Identifier of the account containing the specified attribute.
- **Str** CompanyName – Name of the account, which can be a company name or person's first and last names.

Example

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
  <methodName>Execute</methodName>
  <params>
   <param>
     <value>
      <struct>
       <member>
             <name>Server</name>
             <value>
              <string>BM</string>
            </value>
       </member>
       <member>
        <name>Method</name>
         <value>
         <string>GetAccountListByCustomAttribute_API</string>
         </value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
                <!-- Attribute ID -->
                <value><string>BTN</int></value>
                 <!-- Attribute Value -->
                <value><string>16820</int></value>
                 <!-- SortNo -->
                <value><i4>1</i4></value>
           </data>
          </array>
         </value>
       </member>
      </struct>
     </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
```

```
          <array>
           <data>
            <value>
             <array>
              <data>
                          <!-- Account ID -->
               <value><i4>1000001</i4></value>
                          <!-- Company Name -->
                    <value><string>John Mills</string></value>
                  </data>
                 </array>
                </value>
                <value>
                 <array>
                  <data>
                          <!-- Account ID -->
                    <value><i4>1000002</i4></value>
                          <!-- Company Name -->
                    <value><string>Paul Tall</string></value>
                  </data>
                 </array>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

GetAuthCodeByDomainName_API

Description	Parameters	
	Passed	Returned
Method returns customer authorization code to be used as transfer key.	1	1

Syntax

```
ItemResult EPNIC::GetAuthCodeByDomainName_API(
  Str(255) FullDomainName.
)
returns
  Str(40) AuthCode - customer EPNIC authorization code.
```

Special Notes

Str(255) FullDomainName - domain name + TLD, without www prefix. For example, "domain.com".

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
<methodName>Execute</methodName>
 <params>
  <param>
    <value>
     <struct>
      <member>
       <name>Container</name>
        <value>EPNIC_Container</value>
         </member>
          <member>
           <name>Object</name>
            <value>EPNIC_Object</value>
             </member>
              <member>
               <name>Method</name>
                <value>GetAuthCodeByDomainName_API</value>
                 </member>
                 <member>
                  <name>Params</name>
                  <value>
                   <array>
                    <data>
                     <!-- FullDomainName -->
                     <value>domain.com</value>
                   </data>
                  </array>
                 </value>
                </member>
               </struct>
              </value>
             </param>
            </params>
           </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <!-- Auth Code -->
                <value><string>8950683ff220</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetBasketPrices_API

Description	Parameters	
	Passed	Returned
Method returns online store basket prices. Prices are returned per respective line. Taxes are returned in the extra line with ID=-1.	Depends on order structure	8

Syntax

```
ListResult BM::GetBasketPrices_API(
  Int VendorAccountID;
  Int ProvisioningItemsCounter;
     Str ProvisioningItem_0;
     …
     Str ProvisioningItem_N;
  Int ProvisioningDataSlotCounter;
```

```
     Int ProvisioningItemID_0;
            Int ProvisioningParametersCounter;
                Str ProvisioningParameter_0;
                …
                Str ProvisioningParameter_N;
     …
     Int ProvisioningItemID_N;
            Int ProvisioningParametersCounter;
                Str ProvisioningParameter_0;
                …
                Str ProvisioningParameter_N;
  Int ContactDataCounter;
     Str ContactDataSlot_0;
     …
     Str ContactDataSlot_N;
  Str PromoCodeID.
)
returns
  Int LineID – ID of basket line.
  Double Discount – the line discount amount. If a promotion code has been passed,
then this parameter returns the boolean value in the LineID=-4. This value
specifies the promotion code validation result: 1 - the promo code is valid, in
this case the SKU parameter returns the success message in the LineID=-4; 0 - the
promo code is invalid, in this case the SKU parameter returns the error message
in the LineID=-4.
  Double ExtendedPrice – line total amount.
  Double TotalTax – line tax total (included tax + extra tax). Returned in the
LineID=-1.
  Double ExtraTax – extra tax sum (taxes not included into price). Returned in the
LineID=-1.
  Double SetupPrice – line setup fee.
  Str SKU - the SKU assigned to the line. If a promotion code has been passed,
then this parameter returns the promo code validation result. Success message:
"The promotion code "<promo code>" has been applied successfully." or one of the
error messages: either "The promotion code "<promo code>" is valid, but it is not
enabled for the products selected." or "The promotion code "<promo code>" is
invalid. Please check and correct it or use another promotion code.".
  Double Deposit- pre-paid deposit amount used to top up the customer account
balance.
```

Special Notes

- **Int** VendorAccountID – vendor account ID;
- **Int** ProvisioningItemsCounter - number of provisioning items in order. One order item is created for each service plan / additional resource ordered.
- **Str** ProvisioningItem_N – definition of provisioning item #N in specific format. Provisioning Item is submitted as a string of the following format:

  <PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>

  For ordered resources the string looks like the following:

  <ResourceRateID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>=<Amount>=RESOURCE

  Where:

  - <PlanID> is an ID of ordered service plan;

  - <ResourceRateID> is an ID of service plan resource rate for ordered resource;
  - <PlanPeriodID> is an ID of service plan subscription period;
  - <ProvisioningItemID> is an arbitrary positive unique ID of the provisioning Item inside the order;
  - <ParentItemID> is an identifier of parent provisioning item. Parent provisioning items usually need to be specified for domain provisioning item. ID of hosting provisioning item serves as parent item ID in this case. Pass '-1' for other order item types (hosting service plan, resources);
  - <Amount> is amount of resource ordered (integer);
- **Int** ProvisioningDataSlotCounter - number of total data slots in provisioning parameters section of order. Takes into account all values passed up to the ContactDataCounter;
- **Int** ProvisioningItemID_N - ID of provisioning item #N with which the following provisioning parameters are associated.
- **Int** ProvisioningParametersCounter - number of provisioning parameters associated with current provisioning item;
- **Str** ProvisioningParameter_N - provisioning parameter #N of order item #N. Provisioning parameter is an info required to deliver a service. For example, it can be a domain name in case of ordering domain name, or size in case of ordering shoes. Provisioning parameter is submitted as a string of the following format:

  <ParameterName>=<ParameterValue>

  Where:

  - <ParameterName> is a name of the parameter defined under **Required Parameters** tab of respective service template;
  - <ParameterValue> is a value of the parameter;
- **Int** ContactDataCounter - number of contact data slots;
- **Str** ContactDataSlot_N - data slot #N of contact information on customer placing the order. Identifies customer account on behalf of which the order is placed. Contact data are submitted as strings of the following format:

  <ContactDataSlotName>=<ContactDataSlotValue>

  Where:

  - <ContactDataSlotName> is a name of the Contact Data slot;
  - <ContactDataSlotValue> is a value of the Contact Data slot;

  PBA accepts the following list of Contact Data slot names:

  - LoginID – customer's login to CP;
  - AccountID – ID of customer's account;
- **Str** PromoCodeID – promotion code applied to the order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>BM</value>
        </member>
         <member>
          <name>Method</name>
           <value>GetBasketPrices_API</value>
            </member>
             <member>
              <name>Params</name>
               <value>
                <array>
                 <data>
                  <!-- VendorAccountID -->
                  <value><i4>1</i4></value>
                  <!-- ProvisioningItemsCounter -->
                  <value><i4>2</i4></value>
                  <!-- ProvisioningItem #0. PlanID=1, PlanPeriodID=3,
                 ProvisioningItemID=1, ParentItemID is not defined -->
                  <value>1=3=1=-1</value>
                  <!-- ProvisioningItem #1. ResourceRateID=12,
                 PlanPeriodID=3, ProvisioningItemID=2, ParentItemID=1,
                 Amount=100 -->
                  <value>12=3=2=1=100=RESOURCE</value>
                  <!-- ProvisioningDataSlotCounter -->
                  <value><i4>3</i4></value>
                 <!-- ProvisioningItemID #0. -->
                  <value><i4>0</i4></value>
                  <!-- ProvisioningParametersCounter for
                    the ProvisioningItem #0 -->
                  <value><i4>1</i4></value>
                 <!-- ProvisioningParameter #0.
                      Hosting name is not defined -->
                  <value>DomainID=</value>
                  <!-- ContactDataCounter -->
                  <value><i4>2</i4></value>
                  <!-- Login to CP -->
                  <value>LoginID=johnsmith</value>
                  <!-- Customer's Account ID -->
                  <value>AccountID=1000002</value>
                  <!-- Promo Code ID -->
                  <value><string>PROMO123</string></value>
                 </data>
                </array>
               </value>
              </member>
             </struct>
            </value>
           </param>
          </params>
         </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value><array><data>
                 <!-- Line ID, '1' declares prices and discounts line -->
                 <value><i4>1</i4></value>

                 <!-- Discount -->
                 <value><double>362.500000</double></value>

                 <!-- Extended price -->
                 <value><double>362.500000</double></value>

                 <!-- Total tax -->
                 <value><double>0.000000</double></value>

                 <!-- Extra tax -->
                 <value><double>0.000000</double></value>

                 <!-- Setup price -->
                 <value><double>2.500000</double></value>

                 <!--SKU-->
                 <value><string>SRV8200</string></value>

                 <!--Deposit-->
                 <value><double>0.000000</double></value>
               </data></array></value>
                    <value><array><data>

                      <!-- Line ID, '-1' declares TAXES line -->
                 <value><i4>-1</i4></value>

                 <!-- Discount -->
                 <value><double>0.000000</double></value>

                 <!-- Extended price -->
                 <value><double>0.000000</double></value>

                 <!-- Total tax -->
                 <value><double>65.200000</double></value>

                 <!-- Extra tax -->
                 <value><double>65.200000</double></value>

                 <!-- Setup price -->
                 <value><double>0.000000</double></value>

                 <!--SKU-->
                 <value><string></string></value>

                 <!--Deposit-->
                 <value><double>0.000000</double></value>
                    </data></array></value>
                    <value><array><data>
                 <!--Line ID, '-4' declares Promotion code validation result-->
                 <value><i4>-4</i4></value>
                 <!-- Discount -->
                 <value><double>1.000000</double></value>
                 <!-- Extended price -->
                 <value><double>0.000000</double></value>
                 <!-- Total tax -->
                 <value><double>0.000000</double></value>
                 <!-- Extra tax -->
                 <value><double>0.000000</double></value>
                 <!-- Setup price -->
                 <value><double>0.000000</double></value>
                 <!-- SKU -->
                 <value><string>The promotion code &quot;PROMO123&quot; has been
applied successfully.</string></value>
                 <!--Deposit-->
                 <value><double>0.000000</double></value>
                    </data></array></value>
             </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetCurrencyIDforStore_API

Description	Parameters	
	Passed	Returned
Method returns currency ID for specified online store.	1	1

Syntax

```
ItemResult STORES::GetCurrencyIDforStore_AP (
      Int StoreID
)
returns
      Str CurrencyID - store owner's currency.
```

Special Notes

- **Int** CurrencyIDVendorAcco – store owner's currency ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
   <param>
     <value>
      <struct>
       <member>
              <name>Server</name>
              <value>
               <string>STORES</string>
             </value>
       </member>
       <member>
        <name>Method</name>
         <value>
          <string>GetCurrencyIDforStore_API</string>
         </value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
            <!-- StoreID -->
                <value><i4>1</i4></value>
           </data>
          </array>
         </value>
       </member>
      </struct>
     </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <value>
                <!-- CurrencyID -->
```

```
                 <string>USD</string>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

GetCustomerSubscriptionList_API

Description	Parameters	
	Passed	Returned
This method returns the list of subscriptions belonging to a particular customer. Note: Details are returned for all a customer's subscriptions in any statuses.	2	12

Syntax

```
ItemResult BM::GetCustomerSubscriptionList_API(
  Int AccountID;
  Int SortNo.
)
returns
Int SubscriptionID;
Str SubscriptionName;
Int PlanID;
Str PlanName;
Int PlanPeriodID;
Int Period;
Int PeriodType;
Int StartDate;
Int ExpirationDate;
Int Status;
Int ServStatus;
Str ContainerName.
```

Parameters Description

Input parameters:

- **Int** AccountID – The identifier of a customer for whom you want to see the list of subscriptions owned.
- **Int** SortNo – This parameter is an optional parameter defining how the output data is sorted. The parameter is set in the following way (in this example, N is the number of a column):
  - N – The data is sorted by the N column in ascending order.
  - - N – The data is sorted by the N column in descending order.

Output parameters:

An array of the following subscription data:

- **Int** SubscriptionID – The identifier of a customer's subscription.
- **Str** SubscriptionName – The name of the subscription.

- **Int** PlanID – The identifier of the service plan to which the subscription belongs.
- **Str** PlanName – The service plan name.
- **Int** PlanPeriodID – The identifier of the Subscription Period applied to this subscription.
- **Int** Period – The Subscription Period duration (for example, 1 if the subscription period is 1 year or month).
- **Int** PeriodType – The type of the Subscription Period. This parameter can have the following values:
  - 0 – Hour(s)
  - 1 – Day(s)
  - 2 – Month(s)
  - 3 – Year(s)
- **Int** StartDate – The date of the subscription start in the Unix date format.
- **Int** ExpirationDate – The expiration date of the subscription in the Unix date format.
- **Int** Status – The subscription status. This parameter can have the following values:
  - 10 – Ordered
  - 15 – Trial
  - 30 – Active
  - 40 – Graced
  - 50 – Expired
  - 60 – Terminated
  - 70 – Canceled
  - 80 – Administrative Hold
  - 85 – Credit Hold
  - 89 – Credit and Administrative Hold
- **Int** ServStatus – The subscription service status. This parameter can have the following values:
  - 10 – Not Provisioned
  - 20 – Provisioning
  - 30 – Stopped
  - 40 – Starting
  - 50 – Running
  - 60 – Stopping
  - 70 – Removing
  - 80 – Changing Plan

    - 90 – Removed
- **Str** ContainerName – The name of the service gate for the subscription service

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request -->
<methodCall>
        <methodName>Execute</methodName>
         <params>
          <param>
           <value>
            <struct>
             <member>
              <name>Server</name>
              <value>BM</value>
             </member>
             <member>
              <name>Method</name>
              <value>GetCustomerSubscriptionList_API</value>
             </member>
             <member>
              <name>Params</name>
              <value>
               <array>
                <data>
                 <!-- AccountID -->
                 <value>
                  <i4>1000007</i4>
                 </value>
                 <!-- SortNo -->
                 <value>
                  <i4>1</i4>
                 </value>
                </data>
               </array>
              </value>
             </member>
            </struct>
           </value>
          </param>
         </params>
</methodCall>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
       <params>
        <param>
         <value>
          <struct>
           <member>
            <name>Result</name>
            <value>
             <array>
              <data>
               <value>
                <array>
                 <data>
```

```
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value>
          <i4>1000026</i4>
         </value>
         <!-- SubscriptionName -->
         <value>
          <string>Dummy_plan1</string>
         </value>
         <!-- PlanID -->
         <value>
          <i4>2</i4>
         </value>
         <!-- PlanName -->
         <value>
          <string>Dummy_plan1</string>
         </value>
         <!-- PlanPeriodID. -->
         <value>
          <i4>2</i4>
         </value>
         <!-- Period -->
         <value>
          <i4>1</i4>
         </value>
         <!-- PeriodType -->
         <value>
          <i4>2</i4>
         </value>
         <!-- StartDate -->
         <value>
          <i4>1397419200</i4>
         </value>
         <!-- ExpirationDate -->
         <value>
          <i4>1400011200</i4>
         </value>
         <!-- Status -->
         <value>
          <i4>30</i4>
         </value>
         <!-- ServStatus -->
         <value>
          <i4>50</i4>
         </value>
         <!-- ContainerName -->
         <value>
          <string>DUMMYGATE</string>
         </value>
        </data>
       </array>
      </value>
     </data>
    </array>
   </value>
  </data>
 </array>
</value>
```

```
           </member>
          </struct>
         </value>
        </param>
       </params>
</methodResponse>
```

GetDependanceCategories_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of sales categories assigned to specified service plan as up-sale categories.	2	3

Syntax

```
ListResult BM::GetDependanceCategories_API(
  Int PlanID;
  Int SortNo.
)
returns
  Int CategoryID - sales category ID;
  Str(60) Name - sales category name;
  Int ChildLimit - maximum number of up-sale subscriptions set for up-sale
category (Maximum Up-sale Subscriptions parameter).
```

Special Notes

- **Int** PlanID - service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Category ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
        <value>BM</value>
        </member>
         <member>
          <name>Method</name>
           <value>GetDependanceCategories_API</value>
            </member>
             <member>
             <name>Params</name>
              <value>
              <array>
               <data>
                <!-- PlanID -->
                <value><i4>3</i4></value>
                <!-- SortNo -->
                 <value><i4>1</i4></value>
                 </data>
                </array>
               </value>
              </member>
             </struct>
            </value>
           </param>
          </params>
         </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>
                <!-- Sales Category name -->
                 <value><string>
                  Domain Registration
                 </string></value>
                 <!-- Child Limit -->
                 <value><i4>1023</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
  </param>
 </params>
</methodResponse>
```

GetFullDomainZoneList_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of supported domain zones for specified account.	2	4

Syntax

```
ListResult DOMAINGATE::GetFullDomainZoneList_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int ServiceTemplateID - service template ID;
  Str DomainZoneID - ID of domain zone;
  Str Container - respective registrar plug-in name;
  Int Selected by Default - signifies whether domain zone is selected by default
("1") when customer registers domain is store / CP. Configured under domain
service template properties.
```

Special Notes

- **Int** AccountID - vendor account ID (provider or reseller);
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (ServiceTemplate ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
        <value>DOMAINGATE</value>
         </member>
          <member>
           <name>Method</name>
            <value>GetFullDomainZoneList_API</value>
             </member>
              <member>
               <name>Params</name>
                <value>
                 <array>
                  <data>
                   <!-- AccountID -->
                    <value><i4>1</i4></value>
                     <!-- SortNo -->
                      <value><i4>1</i4></value>
                      </data>
                     </array>
                    </value>
                   </member>
                  </struct>
                 </value>
                </param>
               </params>
              </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
```

```
              <array>
               <data>
                 <!-- Service Template ID -->
                 <value><i4>1000001</i4></value>
                <!-- Domain Zone ID -->
                 <value><string>biz</string></value>
                <!-- Container -->
                 <value><string>MIT</string></value>
                 <!-- Selected by Default -->
                 <value><i4>0</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                 <!-- Service Template ID -->
                 <value><i4>1000002</i4></value>
                 <!-- Domain Zone ID -->
                 <value><string>se</string></value>
                 <!-- Container -->
                 <value><string>IIS</string></value>
                 <!-- Selected by Default -->
                 <value><i4>0</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetHostingSubscriptionList_API

Description	Parameters	
	Passed	Returned
Method returns hosting subscriptions for specified account. **Note:** the method does not return subscriptions that are in the "Terminated", "Cancelled" or "Suspended" statuses.	2	12

Syntax

```
ListResult BM::GetHostingSubscriptionList_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int SubscriptionID - subscription ID;
  Str SubscriptionName - subscription name;
  Int PlanID - ID of subscription service plan;
  Str PlanName - subscription service plan name;
  Int PlanPeriodID - subscription period ID;
  Int Period - subscription period duration (for example, 1 if subscription period
is 1 year);
  Int PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 -
Month(s), 3 - Year(s);
  Int StartDate -  the date of the subscription start in Unix date format;
  Int ExpirationDate - expiration date of the subscription in Unix date format;
  Int Status - subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40
- "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 -
"Administrative Hold", 85 - "Credit Hold", 89 - "Credit and  Administrative
Hold");
  Int ServStatus - subscription service status: 10 - Not Provisioned, 20 -
Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 - Stopping, 70 -
Removing, 80 - Changing Plan, 90 - Removed;
  Str ContainerName - name of service gate subscription service is provisioned
through. Hosting is provisioned through PEMGATE, DUMMYGATE and RESELLERGATE only.
```

Special Notes

- **Int** AccountID - customer account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (SubscriptionID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
        </member>
         <member>
          <name>Method</name>
           <value>GetHostingSubscriptionList_API</value>
            </member>
             <member>
              <name>Params</name>
               <value>
                <array>
                 <data>
                  <!-- AccountID -->
                  <value><i4>1000023</i4></value>
                  <!-- SortNo -->
                  <value><i4>1</i4></value>
                 </data>
                </array>
               </value>
              </member>
             </struct>
            </value>
           </param>
          </params>
         </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
```

```
                <array>
                 <data>
                   <!-- Subscription ID -->
                   <value><i4>1000009</i4></value>
                   <!-- Subscription Name -->
                   <value><string>ID1000009</string></value>
                  <!-- Plan ID -->
                   <value><i4>3</i4></value>
                  <!-- Plan Name -->
                   <value><string>Linux Basic</string></value>
                 <!-- Plan Period ID -->
                   <value><i4>3</i4></value>
                  <!-- Period Duration -->
                   <value><i4>1</i4></value>
                   <!-- Period Type -->
                   <value><i4>3</i4></value>
                   <!-- Start Date -->
                   <value><i4>1245333405</i4></value>
                   <!-- Expiration Date -->
                   <value><i4>1276869405</i4></value>
                  <!-- Subscription Status -->
                   <value><i4>30</i4></value>
                  <!-- Subscription Service Status -->
                   <value><i4>50</i4></value>
                 <!-- Container Name -->
                   <value><string>DUMMYGATE</string></value>
                  </data>
                 </array>
                </value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </member>
        </struct>
       </value>
      </param>
   </params>
  </methodResponse>
```

GetLastSubscrStartDate_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the date of last start date of specified subscription	1	1

Syntax

```
ItemResult BM :: GetLastSubscrStartDate_API (
       Int SubscriptionID.
)
```

```
returns
       Int LastSubscrStartDate - the date when specified subscription was last
started in Unix date format.
```

Special Notes

Int SubscriptionID - ID of subscription for which the last start date is requested

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>GetLastSubscrStartDate_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                                <!-- SubscriptionID -->
                  <value><i4>1000045</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
```

```
                    <array>
                      <data>
                        <value>
                                <!-- LastSubscrStartDate -->
                          <i4>1347403758</i4>
                        </value>
                      </data>
                           </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

GetLastSubscrStopDate_API

Description	Parameters	
	Passed	Returned
Method returns the date of last stop date of specified subscription.	1	1

Syntax

```
ItemResult BM :: GetLastSubscrStopDate_API (
       Int SubscriptionID.
)
returns
       Int LastSubscrStopDate - the date when specified subscription was last
stopped in Unix date format.
```

Special Notes

Int SubscriptionID - ID of subscription for which the last stop date is requested.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
```

```
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>GetLastSubscrStopDate_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                           <!-- SubscriptionID -->
                  <value><i4>1000046</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value>
                           <!-- LastSubscrStopDate -->
                          <i4>1347403758</i4>
                        </value>
                      </data>
                           </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

GetLastToServStatusTransitionDate_API

Description	Parameters	
	Passed	**Returned**
Method returns the last date when requested subscription was in specified service status.	2	1 (or more if erroneous data are requested)

Syntax

```
ItemResult BM :: GetLastToServStatusTransitionDate_API (
       Int SubscriptionID;
       Int ServStatus.
)
returns
       Int LastToServStatusTransitionDate - the date when subscription was last in
specified service status (in Unix date format).
```

Special Notes

- **Int** SubscriptionID - ID of subscription for which the last date in specified service status is requested;
- **Int** ServStatus - subscription service status: 10 - Not Provisioned, 20 - Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 - Stopping, 70 - Removing, 80 - Changing Plan, 90 - Removed.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
           <value>GetLastToServStatusTransitionDate_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                            <!-- SubscriptionID -->
                  <value><i4>1000004</i4></value>
                            <!-- ServStatus -->
                  <value><i4>70</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value>
                                    <!-- LastToServStatusTransitionDate -->
                         <i4>1346962572</i4>
                            </value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse
```

GetLoginSettings_API

Description	Parameters	
	Passed	**Returned**
Method returns login and password settings of specified vendor. Login and password settings are configured under the **System** > **Settings** > **Security** > **Login Settings** submenu of the Navigation tree.	1	7

Syntax

```
ItemResult BM::GetLoginSettings_API(
  Int AccountID.
)
returns
  Int LoginMinLength - signifies the minimal length of the login (symbols);
  Int LoginMaxLength - signifies the maximal length of the login (symbols);
  Int LoginNumbersAllowed - signifies whether digits are allowed in login: "0" -
digits are not allowed, "1" - digits are allowed;
  Int LoginSpecialCharAllowed - signifies whether special characters are allowed
in login: "0" - special characters are not allowed, "1" - special characters are
allowed;
  Str LoginAlert - error message displayed to customer when incorrect login is
submitted;
  Str ManualLoginMask - the mask used to specify not allowed characters in login;
  Int PwdQualityLevel - signifies the password quality level: "0" - None; "1" -
Low; "2" - Medium; "3" - Medium-High; "4" - High; "5" - Very-High.
```

Special Notes

Int AccountID - vendor account ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
       <value>BM</value>
        </member>
         <member>
          <name>Method</name>
           <value>GetLoginSettings_API</value>
            </member>
             <member>
              <name>Params</name>
               <value>
                <array>
                 <data>
                  <!-- AccountID -->
                  <value><i4>1</i4></value>
                </data>
               </array>
              </value>
             </member>
            </struct>
           </value>
          </param>
         </params>
        </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- LoginMinLength -->
              <value><i4>5</i4></value>
              <!-- LoginMaxLength -->
              <value><i4>20</i4></value>
              <!-- LoginNumbersAllowed -->
              <value><i4>1</i4></value>
              <!-- LoginSpecialCharAllowed -->
              <value><i4>1</i4></value>
              <!-- LoginAlert -->
              <value><string>
               Login Name is incorrect. Please make sure that your
               Login Name is 5 to 20 characters long and does not
               contain special or national characters
              </string></value>
              <!-- ManualLoginMask -->
              <value><string>
               ^[A-Za-z0-9]{1}[A-Za-z0-9_\.\-]{0,255}$
              </string></value>
              <!-- PwdQualityLevel -->
              <value><i4>3</i4></value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
   </methodResponse>
```

GetMaxSupportedTransferPlanPeriodDetails_API

Description	Parameters	
	Passed	Returned
Method returns subscription period details of specified service plan, that matches for transfer at most.	2	15

Syntax

```
ItemResult DOMAINGATE::GetMaxSupportedTransferPlanPeriodDetails_API(
  Int PlanID;
  Int VendorID.
  )
returns
  Int PlanPeriodID - service plan subscription period ID;
  Int PlanID - service plan ID;
  Int Period - subscription period duration (for example, 1 if subscription period
is 1 year);
  Int PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 -
Month(s), 3 - Year(s);
  Int Trial - whether subscription period is trial (free): 0 - No, 1 - Yes;
  Double SetupFee - subscription setup fee. It is initial and one time charge
which is taken for subscription for this plan;
  Double SubscriptionFee - subscription recurring fee. It is taken once a billing
period;
  Double RenewalFee - subscription renewal fee. It is charged for subscription
renewal;
```

```
  Double TransferFee - subscription transfer fee. It is charged for domain
transfer and relevant for domain plans only;
  Double NonRefundableAmt - subscription non-refundable amount. The sum is not
refunded to customer when refund is claimed (on subscription cancellation, for
example);
  Int RefundPeriod - subscription refund period duration in days (for example, 5
if refund period is 5 days). During refund period customer can cancel his
subscription and get all money back, except non-refundable amount;
  Int Enabled - whether subscription period is active : 0 - No, 1 - Yes;
  Double NumberOfPeriods - number of billing periods within subscription period.
This parameter is calculated automatically and depends on Billing Period Type
parameter of service plan. For example, if Billing Period Type is "Fixed Number of
Months" and subscription period is 2 years long, there will be 24 billing
periods, meaning customer will be charged recurring fee 24 times;
  Str(1024) TransferFeeText - text displayed as description of transfer fee, for
example ".com Transfer Fee".
  Int SortNumber - order in which subscription periods are shown in online store
and CP: "1" - on top, "2" - below it, etc.
```

Special Notes

- **Int** PlanID - ID of service plan;
- **Int** VendorID - ID of vendor account (provider or reseller).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetMaxSupportedTransferPlanPeriodDetails_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>1</i4></value>
        <!-- VendorID -->
         <value><i4>1</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Plan period ID -->
              <value><i4>1</i4></value>
              <!-- Plan ID -->
              <value><i4>1</i4></value>
             <!-- Period -->
              <value><i4>1</i4></value>
              <!-- Period Type -->
              <value><i4>3</i4></value>
              <!-- Trial -->
              <value><i4>0</i4></value>
              <!-- Setup Fee -->
              <value><double>11.000000</double></value>
              <!-- Subscription Fee -->
              <value><double>11.000000</double></value>
              <!-- Renewal Fee -->
              <value><double>11.000000</double></value>
              <!-- Transfer Fee -->
              <value><double>11.000000</double></value>
              <!-- NonRefundable Amount -->
              <value><double>11.000000</double></value>
              <!-- Refund Period -->
              <value><i4>1</i4></value>
              <!-- Enabled -->
              <value><i4>1</i4></value>
              <!-- Number of Periods -->
              <value><double>12.000000</double></value>
              <!-- Transfer Fee Text -->
              <value><string>
               @@Subscription_Name@ Transfer
               </string></value>
              <!-- Sort Number -->
              <value><i4>8</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
```

```
</methodResponse>
```

GetSharedHostingSubscriptionList_API

	Parameters	
Description	**Passed**	**Returned**
Method returns POA hosting (which are based on PEMGATE container) subscriptions for specified account. **Note:** the method does not return subscriptions that are in the "Terminated" or "Cancelled" statuses.	2	12

Syntax

```
ListResult BM::GetSharedHostingSubscriptionList_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int SubscriptionID - subscription ID;
  Str SubscriptionName - subscription name;
  Int PlanID - ID of subscription service plan;
  Str PlanName - subscription service plan name;
  Int PlanPeriodID - subscription period ID;
  Int Period - subscription period duration (for example, 1 if subscription period
is 1 year);
  Int PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 -
Month(s), 3 - Year(s);
  Int StartDate -  the date of the subscription start in Unix date format;
  Int ExpirationDate - expiration date of the subscription in Unix date format;
  Int Status - subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40
- "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 -
"Administrative Hold", 85 - "Credit Hold", 89 - "Credit and  Administrative
Hold");
  Int ServStatus - subscription service status: 10 - Not Provisioned, 20 -
Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 - Stopping, 70 -
Removing, 80 - Changing Plan, 90 - Removed;
  Str ContainerName - name of service gate subscription service is provisioned
through. Shared hosting is provisioned through PEMGATE only.
```

Special Notes

- **Int** AccountID - customer account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (SubscriptionID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetSharedHostingSubscriptionList_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000023</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Subscription ID -->
                 <value><i4>1000038</i4></value>
                 <!-- Subscription Name -->
                 <value><string>mysubscription</string></value>
                <!-- Plan ID -->
                 <value><i4>18</i4></value>
                 <!-- Plan Name -->
                 <value><string>Linux Basic</string></value>
                 <!-- Plan Period ID -->
                 <value><i4>31</i4></value>
                 <!-- Period duration -->
                 <value><i4>1</i4></value>
                 <!-- Period Type -->
                 <value><i4>3</i4></value>
                 <!-- Start Date -->
                 <value><i4>1219708800</i4></value>
                <!-- Expiration Date -->
                 <value><i4>1251244800</i4></value>
                 <!-- Status -->
                 <value><i4>30</i4></value>
                 <!-- Service Status -->
                 <value><i4>30</i4></value>
                 <!-- Container Name -->
                 <value><string>PEMGATE</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetSubscriptionsListByOrder_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of subscriptions for provisioning items of the specified sales order.	2	1

Syntax

```
ListResult BM::GetSubscriptionsListByOrder_API(
  Int OrderID;
  Int SortNo.
)
returns
  Int SubscriptionID - ID of subscription for a provisioning item of the requested
sales order.
```

Special Notes

Int OrderID - sales order ID;

Int SortNo - any integer. The parameter is required, but not used.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetSubscriptionsListByOrder_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OrderID -->
         <value><i4>16</i4></value>
         <!-- SortNo -->
         <value><i4>0</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Subscription ID for order's
       the 1st provisioning item -->
                 <value><i4>1000009</i4></value>
                </data>
               </array>
              </value>
              <value>
               <array>
                <data>
                 <!-- Subscription ID for order's
       the 2d provisioning item -->
                 <value><i4>1000009</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

GetSubscrTerminateDate_API

Description	Parameters	
	Passed	**Returned**
Method returns the termination date of specified subscription.	1	1

Syntax

```
ItemResult BM :: GetSubscrTerminateDate_API (
Int SubscriptionID.
)
returns
Int SubscrTerminateDate - the date when specified subscription was terminated (in
Unix date format). If the active subscription ID is passed, then the error
message is returned: Termination date is not specified for Subscription
<SubscriptionID>.
```

Special Notes

Int SubscriptionID - ID subscription for which termination date is requested.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>GetSubscrTerminateDate_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                          <!-- SubscriptionID -->
                  <value><i4>1000047</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
```

```
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value>
                                        <!-- SubscrTerminateDate -->
                          <i4>1347403758</i4>
                        </value>
                      </data>
                          </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

GetObjAttrList_API

Description	Parameters	
	Passed	**Returned**
Method returns list of attributes for specified account or user.	3	3

Syntax

```
ListResult BM::GetObjAttrList_API(
  Int ObjectType;
  Int ObjID;
  Int SortNo.
)
returns
  Str AttributeID - ID of attribute;
  Str Value - attribute value. If attribute is not assigned to specified account or
account ID is not passed, method returns empty value;
  Str Tags - attribute tags. COMP, PERS are system predefined tags and refer to the
company and personal accounts, respectively.
```

Special Notes

- **Int** ObjectType - defines type of an object: 0 - available for accounts, 1 - available for users, 2 - available for orders;
- **Int** ObjID - account ID (provider, customer or reseller), user or order ID;

- **Int** SortNo - any integer. The parameter is required, but not used.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetObjAttrList_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ObjectType -->
         <value><i4>0</i4></value>
         <!-- ObjID -->
         <value><i4>123</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <!-- Attribute ID -->
                <value><string>DOB</string></value>
                <!-- Attribute Value -->
                <value><string>13.11.1980</string></value>
                     <!-- Attribute Tags -->
                     <value>
                      <string>COMP, PERS</string>
                     </value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <!-- Attribute ID -->
                <value><string>ICQ</string></value>
                <!-- Attribute Value -->
                <value><string>256894256</string></value>
                     <!-- Attribute Tags -->
                     <value>
                      <string>COMP, PERS</string>
                     </value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetObjAttrListExt_API

Description	Parameters	
	Passed	Returned
Method returns extended (compared to one of GetObjAttrList_API) list of attributes for specified account, user or order.	3	4

Syntax

```
ListResult BM::GetObjAttrListExt_API(
  Int ObjectType;
  Int ObjID;
  Int SortNo.
)
returns
 Str AttributeID - ID of attribute;
 Str TypeOfAttribute - type of attribute (e.g. STR, DATE, INT, etc.);
 Str Value - attribute value. If attribute is not assigned to specified account or
account ID is not passed, method returns empty value;
 Str Tags - attribute tags. COMP, PERS are system predefined tags and refer to the
company and personal accounts, respectively.
```

Special Notes

- **Int** ObjectType - defines type of an object: 0 - available for accounts, 1 - available for users, 2 - available for orders;
- **Int** ObjID - the ID of the account (provider, customer or reseller), user or order;
- **Int** SortNo - any integer. The parameter is required, but not used.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetObjAttrListExt_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ObjectType -->
         <value><i4>0</i4></value>
         <!-- ObjID -->
         <value><i4>123</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
```

```
            <data>
             <value>
              <array>
               <data>

                <!-- Attribute ID -->
                <value><string>ICQ</string></value>

                     <!-- Type of Attribute -->
                <value><string>STR</string></value>

                <!-- Attribute Value -->
                <value><string>256894256</string></value>

                     <!-- Attribute Tags -->
                     <value>
                      <string>COMP, PERS</string>
                     </value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

GetOnlineStoresRewriteRules_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the .htaccess file content for specified online store.	1	2

Syntax

```
ListResult STORES::GetOnlineStoresRewriteRules_API(
  Int StoreID_1;
  ...
  Int StoreID_N.
)
returns
  Int StoreID - ID of proxy online store;
  Str RewriteRulesBody - .htaccess file content (proxy online store itself).
```

Special Notes

- **Int** StoreID - ID of the existing online store. The list of online stores can be accessed from the **Product Director** > **Online Store Manager** > **Stores** submenu of Navigation tree.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetOnlineStoresRewriteRules_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- StoreID -->
         <value><i4>2</i4></value>
         <!-- StoreID -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
```

```
                <array>
                 <data>
                  <!-- Store ID -->
                  <value><i4>2</i4></value>
                  <!-- RewriteRulesBody -->
                  <value><string>
RewriteEngine On
RewriteRule ^%URL_SUFFIX%(.*\.(png|gif|jpg|css|js|ico))$ http://10.23.0.131/$1
[P]
RewriteRule ^%URL_SUFFIX%(.*\/\d+)$ http://10.23.0.131/$1 [P,L]
RewriteRule ^%URL_SUFFIX%$ /%URL_SUFFIX%index.php [R,L]
RewriteRule ^%URL_SUFFIX%index\.php.*
http://10.23.0.131/index.php?StoreID=2&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_STRING
} [P,L]
RewriteRule ^%URL_SUFFIX%http/index\.php.*
http://10.23.0.131/http/index.php?StoreID=2&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_S
TRING} [P,L]
RewriteRule ^%URL_SUFFIX%http/getcss\.php.*
http://10.23.0.131/http/getcss.php?StoreID=2&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_
STRING} [P,L]
RewriteRule ^(%URL_SUFFIX%)?http/blank\.html.* http://10.26.0.231/http/blank.html
RewriteRule
^%URL_SUFFIX%http/(info|rateInfo|hit|sync_conf|terms|americanexpress|visa|getjs).
php.*
http://10.23.0.131/http/$1.php?StoreID=2&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_STRI
NG} [P,L]
RewriteRule ^%URL_SUFFIX%(.*)$ http://10.23.0.131$1 [P]
                  </string></value>
                 </data>
                </array>
               </value>
               <value>
                <array>
                 <data>
                  <!-- Store ID -->
                  <value><i4>1</i4></value>
                  <!-- RewriteRulesBody -->
                  <value><string>
RewriteEngine On
RewriteRule ^%URL_SUFFIX%(.*\.(png|gif|jpg|css|js|ico))$ http://10.23.0.131/$1
[P]
RewriteRule ^%URL_SUFFIX%(.*\/\d+)$ http://10.23.0.131/$1 [P,L]
RewriteRule ^%URL_SUFFIX%$ /%URL_SUFFIX%index.php [R,L]
RewriteRule ^%URL_SUFFIX%index\.php.*
http://10.23.0.131/index.php?StoreID=1&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_STRING
} [P,L]
RewriteRule ^%URL_SUFFIX%http/index\.php.*
http://10.23.0.131/http/index.php?StoreID=1&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_S
TRING} [P,L]
RewriteRule ^%URL_SUFFIX%http/getcss\.php.*
http://10.23.0.131/http/getcss.php?StoreID=1&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_
STRING} [P,L]
RewriteRule ^(%URL_SUFFIX%)?http/blank\.html.* http://10.23.0.131/http/blank.html
RewriteRule
^%URL_SUFFIX%http/(info|rateInfo|hit|sync_conf|terms|americanexpress|visa|getjs).
php.*
http://10.23.0.131/http/$1.php?StoreID=1&amp;subdir=%URL_SUFFIX%&amp;%{QUERY_STRI
NG} [P,L]
RewriteRule ^%URL_SUFFIX%(.*)$ http://10.23.0.131$1 [P]
                  </string></value>
                 </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

GetOrder_API

Description	Parameters	
	Passed	Returned
Method returns details of specified order.	1	18

Syntax

```
ItemResult BM::GetOrder_API(
  Int OrderID - requested order ID.
  )
returns
  Int OrderID - requested order ID;
  Str OrderNumber - requested order number;
  Int VendorAccountID - vendor account ID (provider or reseller);
  Int CustomerID - customer account ID;
  Str OrderStatusID - requested order status;
  Str OrderTypeID - requested order type;
  Int CreationTime - order creation time in Unix format;
  Int OrderDate - order creation date in Unix format;
  Double Total - order total sum, without taxes and discounts;
  Double TaxTotal - total of taxes;
  Double DiscountTotal - total of discount applied to the order;
  Double MerchTotal - order total, including taxes and discount amount;
  Str Comments - free form order description;
  Int ExpirationDate - order expiration date in Unix format;
  Str PromoCode - promotion code applied to the order;
  Str SalesBranchID - sales branch ID;
  Str SalesPersonID - sales person ID.
  Str CurrencyID - three-letter code of the currency  used in requested order.
```

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetOrder_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OrderID -->
         <value><i4>16</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
                                   <!-- OrderID -->
              <value><i4>16</i4></value>
                                   <!-- Order Number -->
              <value><string>SO000008</string></value>

                            <!-- Vendor Account ID-->
              <value><i4>1</i4></value>

                            <!-- Customer Account ID-->
              <value><i4>1000002</i4></value>

                            <!-- Order Status ID-->
              <value><string>WP</string></value>

                            <!-- Order Type ID-->
              <value><string>SO</string></value>

                            <!-- Order Creation Time-->
              <value><i4>1245328433</i4></value>

                            <!-- Order Creation Date-->
              <value><i4>1245268800</i4></value>

                            <!-- Order Total -->
              <value><double>85.000000</double></value>

                            <!-- Order Tax Total -->
              <value><double>0.000000</double></value>

                            <!-- Discount Total -->
              <value><double>0.000000</double></value>

                            <!-- Order MerchTotal -->
              <value><double>85.000000</double></value>

                            <!-- Order Description -->
              <value><string>
               Subscription on Plan #3 (Linux Basic) for 1 Year(s).
              </string></value>

                            <!-- Order Expiration Date-->
              <value><i4>1245528000</i4></value>

                            <!-- Promo Code-->
              <value><string></string></value>

                            <!-- Sales Branch ID-->
              <value><string>2</string></value>

                            <!-- Sales Person ID-->
              <value><string>1</string></value>

                            <!-- CurrencyID-->
              <value><string>USD</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

GetUsersListForAccount_API

Description	Parameters	
	Passed	Returned
The method returns the list of active users (those who are not in the *Disabled* status) for a specified account. **Note**: You can use this method, for instance, when you need to find the *admin* user login for an account.	2	3

Syntax

```
ListResult BM::GetUsersListForAccount_API(
  Int AcountID;
  Int SortNo.
)
returns
  Int UserID;
  Str Login;
  Str FullName.
```

Parameters Description

Input Parameters:

- **Str** AccountID – Identifier of the account, the users of which are to be listed.
- **Int** SortNo – Parameter defining how the output data is sorted. The parameter is set the following way:
  - 1 – The data is sorted by the first column in ascending order.
  - 2 – The data is sorted by the second column in ascending order.
  - -1 – The data is sorted by the first column in descending order.
  - -2 – The data is sorted by the second column in descending order.

Output Parameters:

- **Int** UserID – Identifier of the account user.
- **Str** Login – User login name.
- **Str** FullName – Full name of the user.

Example

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
  <methodName>Execute</methodName>
  <params>
   <param>
     <value>
      <struct>
       <member>
             <name>Server</name>
             <value>
              <string>BM</string>
            </value>
       </member>
       <member>
        <name>Method</name>
         <value>
         <string>GetUsersListForAccount_API</string>
         </value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
                        <!-- Account ID -->
                <value><i4>1000001</i4></value>
                         <!-- SortNo -->
                <value><i4>1</i4></value>
           </data>
          </array>
         </value>
       </member>
      </struct>
     </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
```

```
            <value>
             <array>
              <data>
                           <!-- User ID -->
               <value><i4>1000001</i4></value>
                           <!-- User Login -->
                    <value><string>john</string></value>
                           <!-- User Full Name -->
                    <value><string>John F. Mills</string></value>
                  </data>
                 </array>
                </value>
                <value>
                 <array>
                  <data>
                           <!-- User ID -->
                    <value><i4>1000002</i4></value>
                           <!-- User Name -->
                    <value><string>paul</string></value>
                           <!-- User Full Name -->
                    <value><string>Paul Tall</string></value>
                  </data>
                 </array>
                </value>
                <value>
                 <array>
                   <data>
                           <!-- User ID -->
                     <value><i4>1000003</i4></value>
                           <!-- User Name -->
                     <value><string>oscar</string></value>
                           <!-- User Full Name -->
                    <value><string>Oscar P. Terson</string></value>
                  </data>
                 </array>
                </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

GetVendorCurrency_API

Description	Parameters	
	Passed	**Returned**
Method returns details of the currency used by vendor.	1	4

Syntax

```
ItemResult BM::GetVendorCurrency_API(
  Int VendorAcountID - requested vendor ID.
  )
returns
  Str CurrencyID - three-letter code of the currency (USD, EUR, etc.);
  Str CurrencySymbol - currency sign ($, €, etc.);
  Int CurrencySymbolLocation - location of the currency symbol;
  Int CurrencyPrecision - number of digits after decimal point.
```

Special Notes

Int CurrencySymbolLocation - defines the position of the currency symbol: 1 - at the right of the amount, 2 - at the left of the amount.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>GetVendorCurrency_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1000009</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
                    <!-- CurrencyID -->
              <value><string>USD</string></value>
                    <!-- CurrencySymbol -->
              <value><string>$</string></value>
                    <!-- CurrencySymbolLocation-->
              <value><i4>2</i4></value>
                    <!-- CurrencyPrecision -->
              <value><i4>2</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

IncrementResourceUsage_API

	Parameters	
Description	**Passed**	**Returned**
Method increments the resources usage of specified subscription by amount specified. Resource usage is increased or decreased depending on whether delta amount is positive or negative.	3 or more	1

Syntax

```
ErrorMessage DUMMYGATE::IncrementResourceUsage_API(
  Int SubscriptionID;
  Int ResourceID_1;
  Double DeltaAmount_1;
  ...
  Int ResourceID_N;
  Double DeltaAmount_N.
)
returns
 Str ErrorMessage - message after resource usage has been updated: "Resources on
subscription # have been updated.".
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** ResourceID - resource ID;
- **Double** DeltaAmount - amount resource usage is incremented by.
-

Note: the method is also available in BM container (for backwards compatibility) but starting from PBA-5.2 will be only available on DUMMYGATE.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DUMMYGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>IncrementResourceUsage_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000014</i4></value>
         <!-- ResourceID -->
         <value><i4>10023</i4></value>
         <!-- DeltaAmount -->
         <value><double>-1.0000</double></value>
         <!-- ResourceID -->
         <value><i4>10024</i4></value>
         <!-- DeltaAmount -->
         <value><double>10.0000</double></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Resources on subscription #1000014 have been updated.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

InitResellerDefaults_API

	Parameters	
Description	**Passed**	**Returned**
Method completes initialization of a non-branded reseller (also known as the one-step wizard), i.e. sets the currency passed as an input parameter and completes the initialization. **Important**: It is strongly recommended to avoid bulk creation of resellers, because this may result in a deadlock. When a reseller is created using API, first the PlaceOrderAndAuthorize_API method is called and then the reseller is initiated using the InitResellerDefaults_API method. The resellers creation should be queued: create and initialize first reseller, after this create and initialize the second reseller, and so on.	2	1

Syntax

```
ErrorMessage RESELLERGATE::InitResellerDefaults_API(
  Int AccountID;
  Str Currency;
)
returns
  Str ErrorMessage - message after the specified record has been updated:
"Operation done."
```

Special Notes

- **Int** AccountID - ID of the reseller account to be initialized;
- **Str** Currency - The string representation of the Currency ID to be set up for the reseller.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>RESELLERGATE</value>
          </member>
          <member>
            <name>Method</name>
            <value>InitResellerDefaults_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                  <!--AccountID-->
                  <value><i4>1000042</i4></value>
                  <!--Currency-->
                  <value>USD</value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

LicenseInstallationCompleted_API

Description	Parameters	
	Passed	Returned
Method changes license status to *Installation Failed* or *Installed*, according to POA response.	3	1

Syntax

```
ErrorMessage KAGATE::LicenseInstallationCompleted_API(
       Int SubscriptionID - ID of subscription license is installing to.
       Int CompletionStatus - license installation result, POA response: 0 -
installation failed, 1 - installed.
       Str LicenseType - empty parameter.
)
returns
       Str ErrorMessage - message after license status has been changed: "Operation
done."
```

Special Notes

Str `LicenseType` - the parameter is required for correct method execution. Due to the license type is always passed empty from POA, an empty value must be used. See XML example.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>KAGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>LicenseInstallationCompleted_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
```

```
                    <!-- SubscriptionID -->
         <value><i4>1025</i4></value>
                    <!-- CompletionStatus -->
         <value><i4>1</i4></value>
                    <!-- Required Empty Parameter -->
             <value></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

NameServersChanged_API

	Parameters	
Description	**Passed**	**Returned**
Method updates name servers of specified domain in the internal *SyncNameServers* table. When provider opens the domain properties window, domain current name servers are compared with data in the *SyncNameServers* table. If name servers do not match, the **Synchronize NS** button is displayed in the domain properties window.	2	1

Syntax

```
ErrorMessage DOMAINGATE::NameServersChanged_API(
  Str(255) FullDomainName;
  Str(40) PrimaryNameServer;
  Str(40) SecondaryNameServer;
  Str(40) ThirdNameServer;
  Str(40) FourthNameServer.
)
```

```
returns
  Str ErrorMessage - message on name servers change: "Operation done.".
```

Special Notes

- **Str(255)** FullDomainName - domain name + TLD, without www prefix. For example, "domain.com";
- **Str(40)** PrimaryNameServer - domain primary NS;
- **Str(40)** SecondaryNameServer - domain secondary NS;
- **Str(40)** ThirdNameServer - domain third NS;
- **Str(40)** FourthNameServer - domain fourth NS.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>NameServersChanged_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Full Domain Name -->
         <value>domain.com</value>
         <!-- PrimaryNameServer -->
         <value>ns1.provider.com</value>
         <!-- SecondaryNameServer -->
         <value>ns2.provider.com</value>
         <!-- ThirdNameServer -->
         <value>ns3.provider.com</value>
         <!-- FourthNameServer -->
         <value>ns4.provider.com</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

NotificationDetailsGet_API

	Parameters	
Description	**Passed**	**Returned**
Method retrieves details of notification by given email ID.	1	11

Syntax

```
ListResult MESSAGE::NotificationDetailsGet_API(
  Int AmailID.
  )
returns
  Int MessageID - message ID that corresponds to given e-mail;
  Str ToAccountID - recipient PBA account ID;
  Int FromAccountID - sender PBA account ID;
  Str ToEmail - recipient’s mailbox;
  Str BccEmail - list of mailboxes in BCC section
  Str FromEmail - sender’s mailbox;
  Str RefNumber - reference number;
  Int MessageType - message type (1 – HTML email, 2 – text email, 3 – PDF);
  Str Subject - message subject;
  Str Body - message body;
  Int SendDate - timestamp of corresponding mail delivery.
```

Special Notes

- **Int** AmailID - ID of e-mail being sent.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>MESSAGE</value>
          </member>
          <member>
            <name>Method</name>
            <value>NotificationDetailsGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                    <!-- AmailID -->
                  <value><i4>73</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                                  <!-- MessageID -->
                        <value>
                          <i4>59</i4>
                        </value>
                                  <!-- ToAccountID -->
                        <value>
```

```
                          <i4>1000009</i4>
                        </value>
                                  <!-- FromAccountID -->
                        <value>
                          <i4>1</i4>
                        </value>
                                  <!-- ToEmail -->
                        <value>
                         <string>sch@here.com</string>
                        </value>
                                  <!-- BccEmail -->
                        <value>
                          <string>
                          </string>
                        </value>
                                  <!-- FromEmail -->
                        <value>
                          <string>bill@hsol.com</string>
                        </value>
                                  <!-- RefNumber   -->
                        <value>
                          <string>
                          </string>
                        </value>
                                  <!-- MessageType -->
                        <value>
                          <i4>1</i4>
                        </value>
                                  <!-- Subject -->
                        <value>
                          <string>Order Sales Order SO000034 placed</string>
                        </value>
                                  <!-- Body   -->
                        <value>
                          <string>&lt;HTML&gt;
                                        &lt;BODY&gt;
                                        Order Place Notification &lt;BR /&gt;
                                        With best regards, &lt;BR /&gt;
                                        Provider Inc.
                                        &lt;/BODY&gt;
                                        &lt;/HTML&gt;
                          </string>
                        </value>
                                  <!-- SendDate -->
                        <value>
                          <i4>1337267796</i4>
                        </value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

OrderAttributeByVendorListGet_API

	Parameters	
Description	**Passed**	**Returned**
This method returns the list of *order* attributes assigned to a specified vendor.	2	The number of returned parameters depends on the number of assigned attributes.

Syntax

```
BM::OrderAttributeByVendorListGet_API(
  Int VendorID;
  Int SortNo.
)
returns
  an array of items containing order attribute data:
  Str AttributeID;
  Str Name;
  Str Description;
  Str Type;
  Int Status;
  Int SortNumber.
```

Parameters Description

Input parameters:

- **Int** VendorID - Identifier of the vendor for whom the assigned attributes are to be listed.
- **Int** SortNo - This is an optional parameter defining how the output data is sorted. The parameter is set the following way:
  - `1` - The data is sorted by the first column in ascending order.
  - `2` - The data is sorted by the second column in ascending order.
  - `-1` - The data is sorted by the first column in descending order.
  - `-2` - The data is sorted by the second column in descending order.

Output parameters:

An array of items containing the following order attribute data:

- **Str** AttributeID - Identifier of an attribute assigned to the specified vendor.
- **Str** Name - Name of the attribute.

- **Str** Description - Description of the attribute.
- **Str** Type - Type of the attribute data. The parameter value can be as follows:
  - *INT* - an attribute of this type prompts a customer to enter an integer value.
  - *STR* - an attribute of this type prompts a customer to enter a text up to 80 characters in length. The input box is displayed as a single line in this case.
  - *STRA* - an attribute of this type prompts a customer to enter a text over 80 characters in length. The input box is displayed as a text box.
  - *DATE* - an attribute of this type prompts a customer to enter a date. The calendar icon is displayed next to the input box.
  - *PASSWD* - an attribute of this type prompts a customer to enter a password.
  - *NAME_LOGIN* - an attribute of this type prompts a customer to enter a login name.
  - *DOMAIN_NAME* - an attribute of this type prompts a customer to enter a domain name.
  - *STRASCII* - an attribute of this type prompts a customer to enter a text in the ASCII encoding. It is useful when entering the contact information: strictly the ASCII format is required.
- **Int** Status - This parameter defines whether specifying the attribute value is to be mandatory or optional. The parameter value can be one of the following:
  - *10* - -if entering the attribute value is optional.
  - *20* - -if entering the attribute value is required.
- **Int** SortNumber - This is an optional parameter defining by which column the output data is sorted.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>OrderAttributeByVendorListGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
<!-- Identifier of the vendor for which you want to get the list of assigned order attributes displayed. -->
                  <value>
                    <i4>1000001</i4>
                  </value>
<!-- Number of column by which you want the output data to be sorted. -->
                  <value>
                    <i4>1</i4>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <value>
             <array>
```

```
              <data>
          <!-- Identifier of an attribute. -->
               <value><string>ATTR1</string></value>
          <!-- Name of the attribute. -->
               <value><string>Attribute1</string></value>
          <!-- Description of the attribute -->
               <value><string>Order attribute 1</string></value>
          <!-- Type of the attribute data. -->
               <value><string>STR</string></value>
          <!-- This parameter defines whether specifying the attribute value is to be mandatory (20) or optional (10). -->
               <value><i4>10</i4></value>
          <!-- This is an optional parameter defining by which column the output data is sorted. -->
               <value><i4>1</i4></value>
              </data>
             </array>
            </value>
            <value>
             <array>
              <data>
               <value><string>ATTR2</string></value>
               <value><string>Attribute2</string></value>
               <value><string>Order attribute2</string></value>
               <value><string>STR</string></value>
               <value><i4>20</i4></value>
               <value><i4></i4></value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

OrderByStatusListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of orders of specified type and status.	4	19

Syntax

```
ListResult BM::OrderByStatusListGet_API(
  Int VendorAccountID;
  Str OrderTypeID;
  Str OrderStatusID;
  Int StartTime;
  Int SortNo.
  )
returns
  Int OrderID - order ID;
  Str OrderNumber - order number;
  Int CustomerID - customer account ID;
  Str CustomerName - customer account name;
  Str OrderStatus - requested order status;
  Str OrderType - requested order type;
  Int CreationTime - order creation time in Unix format;
  Int OrderDate - order creation date in Unix format;
  Double Total - order total sum, without taxes and discounts;
  Double TaxTotal - total of taxes;
  Double DiscountTotal - total of discount applied to the order;
  Double MerchTotal - order total, including taxes and discount amount;
  Int ExpirationDate - order expiration date in Unix format;
  Str PromoCode - promotion code applied to the order;
  Int SalesBranchID - sales branch ID;
  Str SalesBranchName - sales branch name;
  Int SalesPersonID - sales person ID;
  Str SalesPersonName - sales person name;
  Str Comments - free form comments to the order.
```

Special Notes

- **Int** VendorAccountID - vendor account ID (provider or reseller);
- **Str** OrderTypeID - two-letter order type ID. The list of order types can be accessed from the **System** > **Settings** > **Order Processing** > **Order Types** submenu of the Navigation tree;
- **Str** OrderStatusID - two-letter order status ID. The list of order statuses can be accessed from the **Statuses** tab of any order type;
- **Int** StartTime - signifies Unix time stamp. If the parameter is specified, only those orders that came to the status specified in the OrderStatusID parameter at StartTime or later should be returned by the method;

- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (OrderID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>OrderByStatusListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
        <!-- OrderTypeID -->
         <value>SO</value>
         <!-- OrderStatusID -->
         <value>NW</value>
         <!-- Start Time -->
         <value><i4>1254497400</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
              <array>
               <data>
                <!-- Order ID -->
                <value><i4>8</i4></value>
                <!-- Order Number -->
                <value><string>SO000005</string></value>
                <!-- Customer ID -->
                <value><i4>1000001</i4></value>
                <!-- Customer Name -->
                <value><string>John Smith</string></value>
                <!-- Order Status -->
                <value><string>NW</string></value>
                <!-- Order Type -->
                <value><string>SO</string></value>
                <!-- Order Creation Time -->
                <value><i4>1245326674</i4></value>
                <!-- Order Date -->
                <value><i4>1245268800</i4></value>
                <!-- Order Total -->
                <value><double>5.000000</double></value>
                <!-- Order Tax Total -->
                <value><double>0.000000</double></value>
                <!-- Order Discount Total -->
                <value><double>0.000000</double></value>
                <!-- Order Merch Total -->
                <value><double>5.000000</double></value>
                <!-- Order Expiration Date -->
                <value><i4>1245528000</i4></value>
                <!-- Order PromoCode -->
                <value><string> </string></value>
                <!-- Sales Branch ID -->
                <value><i4>12</i4></value>
                <!-- Sales Branch Name -->
                <value><string>Fifth Avenue</string></value>
                <!-- Sales Person ID -->
                <value><i4>8</i4></value>
                <!-- Sales Person Name -->
                <value><string>Paul Braun</string></value>
                <!-- Comments -->
                <value><string>
                 Order for Subscription on Plan #2 (.se (IIS)) for 1 Year(s).
               </string></value>
              </data>
             </array>
            </value>
     ...
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
   </methodResponse>
```

OrderFinDetailsListGet_API

	Parameters	
Description	**Passed**	**Returned**
The method returns the list of details for a specified order. **Note:** In the results returned, the pricing matches the *Extra Precision* configuration in PBA. For more details, refer to the **Extra Precision in PBA Pricing** section in **PBA Provider's Guide**.	2	14

Syntax

```
ListResult BM::OrderFinDetailsListGet_API(
  Int OrderID;
  Int SortNo.
  )
returns
  Int SortNo;
  Int DetailID;
  Str Description;
  Int DetailType;
  Double Quantity;
  Str UOM;
  Double UnitPrice;
  Double DiscountAmount;
  Double ExtendedPrice;
  Str TaxCategory;
  Int Subscription;
  Double Duration;
  Int BillingPeriod;
  Int BillingPeriodType.
```

Parameters Description

Input Parameters:

- **Int** OrderID – Identifier of a sales order;
- **Int** SortNo – Parameter that defines how the output data is sorted. The parameter values and their meanings are as follows:
  - 1 – the output is sorted by first column in ascending order;
  - 0 – the output is sorted by second column in ascending order;
  - -1 – the output is sorted by first column (SortNo) in descending order.

Output Parameters:

- **Int** SortNo – Parameter that defines how the output data is sorted.
- **Int** DetailID – Identifier of the order detail.
- **Str** Description – Description of the order detail.

  Note: The method returns a number of spaces (&amp;#32;) before a subdetail description. This number of preceding spaces depends on the detail nesting level.

- **Int** DetailType – Order detail type. This parameter can have the following values:
  - 0 – unknown.
  - 100 – plan setup.
  - 101 – plan switch.
  - 105 – plan renew.
  - 10100 – plan setup refund.
  - 10110 – plan recurring refund.
  - 10120 – resource setup refund.
  - 110 – plan recurring.
  - 120 – resource setup.
  - 125 – resource switch.
  - 130 – resource recurring.
  - 10130 – resource recurring refund.
  - 140 – resource overusage.
  - 150 – resource downgrade.
  - 155 – resource downgrade (credit later).
  - 400 – discount.
  - 600 – adjustment.
  - 700 – non provisioning item.
  - 799 – bill record.

  - 900 – refund.
- **Double** Quantity – Ordered item quantity.
- **Str** UOM – Ordered item unit of measure.
- **Double** UnitPrice – Price of the ordered item.
- **Double** DiscountAmount – Amount of the discount for the item purchase.
- **Double** ExtendedPrice – Total sum of the item purchase.
- **Str** TaxCategory – Tax category applicable to the item purchase.
- **Int** Subscription – Subscription identifier.
- **Double** Duration – Period of time for which the order item is billed.
- **Int** BillingPeriod – Billing period duration (if BillingPeriodType=2 and BillingPeriod=1, the subscription billing period is 1 month).
- **Int** BillingPeriodType – Billing period type. This parameter can have the following values:
  - 4 – monthly, on statement cycle date.
  - 2 – fixed number of months.
  - 3 – fixed number of years.

Note: This method can return the 3.4028234663852886e+38 value, which means 0.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
         <value>OrderFinDetailsListGet_API</value>
       </member>
       <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <!-- OrderID -->
            <value><i4>1000003</i4></value>
           <!-- SortNo -->
           <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
               <!-- Sort No -->
                <value><i4>1</i4></value>
               <!-- Detail ID -->
                <value><i4>1</i4></value>
               <!-- Detail description -->
                <value><string>mysite.de Setup</string></value>
               <!-- Detail type -->
                <value><i4>100</i4></value>
               <!-- Quantity -->
               <value><double>1.000000</double></value>
               <!-- Detail UOM -->
                <value><string>item</string></value>
               <!-- Unit price -->
                <value><double>1.000000</double></value>
               <!-- Discount amount -->
                <value><double>0.000000</double></value>
               <!-- Extended price -->
                <value><double>1.000000</double></value>
               <!-- Tax category -->
                <value><string>Default </string></value>
               <!-- Subscription ID -->
                <value><i4>1000069</i4></value>
                    <-- Duration -->
                     <value><double>12.000000</double></value>
                    <-- Billing Period -->
                     <value><i4>1</i4></value>
                    <-- Billing Period Type -->
                     <value><i4>4</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
               <!-- Sort No -->
                <value><i4>2</i4></value>
               <!-- Detail ID -->
                <value><i4>2</i4></value>
               <!-- Detail description -->
                <value><string>mysite.de Recurring</string></value>
               <!-- Detail type -->
                <value><i4>110</i4></value>
               <!-- Quantity -->
                <value><double>1.000000</double></value>
               <!-- Detail UOM -->
                <value><string>item</string></value>
               <!-- Unit price -->
                <value><double>1.000000</double></value>
               <!-- Discount amount -->
                <value><double>0.000000</double></value>
               <!-- Extended price -->
                <value><double>1.000000</double></value>
               <!-- Tax category -->
                <value><string>Default</string></value>
               <!-- Subscription ID -->
                <value><i4>1000069</i4></value>
                    <-- Duration -->
                     <value><double>1.000000</double></value>
                    <-- Billing Period -->
                     <value><i4>1</i4></value>
                    <-- Billing Period Type -->
                     <value><i4>4</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

OrderFinDetailsListGetExt_API

Description	Parameters	
	Passed	Returned

Method returns extended (compared to one of the OrderFinDetailsListGet_API) list of details for specified order. **Note:** in the results returned, the pricing matches the Extra Precision configuration in PBA. For more details refer to the *Extra Precision in PBA Pricing* section in *PBA Provider's Guide.*	2	16

Syntax

```
ListResult BM::OrderFinDetailsListGetExt_API(
  Int OrderID;
  Int SortNo.
  )
returns
  Int DetailID - order detail ID;
  Str SKU - order detail SKU;
  Str Description - order detail description;
```

Note: method returns a number of spaces (&amp;#32;) before subdetail description. The number of spaces preceding depends on detail nesting level.

```
  Double Quantity - ordered item quantity;
  Str UOM - ordered item unit of measure;
  Double Duration - period of time for which the order item is billed;
  Str Period - description (name) of the period;
  Str UnitPrice - ordered item price (with the currency stated);
  Str TaxCategory - item tax category;
  Str DiscountAmount - item discount amount(with the currency stated);
  Str ExtendedPrice - item total sum(with the currency stated);
  Int StartDate - start date of the period for which the order item is billed;
  Int EndDate - end date of the period for which the order item is billed;
  Int Subscription - subscription ID;
  Int DetailType - document detail type;
  Str ResourceDescription - description of resource in order details.
```

Special Notes

- **Int** OrderID - sales order ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (SortNo) in descending order.
- **Int** DetailType possible values are:
- 0 – unknown.
- 100 – plan setup.
- 101 – plan switch.
- 105 – plan renew.
- 10100 – plan setup refund.

- 10110 – plan recurring refund.
- 10120 – resource setup refund.
- 110 – plan recurring.
- 120 – resource setup.
- 125 – resource switch.
- 130 – resource recurring.
- 10130 – resource recurring refund.
- 140 – resource overusage.
- 150 – resource downgrade.
- 155 – resource downgrade (credit later).
- 400 – discount.
- 600 – adjustment.
- 700 – non provisioning item.
- 799 – bill record.
- 900 – refund.

Note: This method can return the 3.4028234663852886e+38 value, which means 0.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
         <value>OrderFinDetailsListGetExt_API</value>
       </member>
       <member>
       <name>Params</name>
        <value>
         <array>
          <data>
            <!-- OrderID -->
            <value><i4>1000003</i4></value>
```

```
              <!-- SortNo -->
            <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                          <!-- DetailID -->
               <value><i4>438</i4></value>
                          <!-- SKU -->
                    <value><string></string></value>
                          <!-- Description -->
                    <value><string>License 100 user Setup</string></value>
                          <!-- Quantity -->
                    <value><double>1.000000</double></value>
                          <!-- UOM -->
                    <value><string>\xd1\x88\xd1\x82.</string></value>
                          <!-- Duration -->

       <value><double>340282346638528859811704183484516925440.000000</double></val
ue>
                          <!-- Period -->
                    <value><string></string></value>
                          <!-- UnitPrice -->
                    <value><string>USD 300.0000</string></value>
                          <!-- TaxCategory -->
                    <value><string>Default</string></value>
                          <!-- DiscountAmount -->
                    <value><string>USD 0.0000</string></value>
                          <!-- ExtendedPrice -->
                    <value><string>USD 300.0000</string></value>
                          <!-- StartDate -->
                    <value><i4>1338408000</i4></value>
                          <!-- EndDate -->
                    <value><i4>1338408000</i4></value>
                          <!-- Subscription -->
                    <value><i4>1000084</i4></value>
                          <!-- DetailType -->
                    <value><i4>100</i4></value>
                          <!-- ResourceDescription -->
                    <value><string>some plan</string></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                          <!-- DetailID -->
               <value><i4>439</i4></value>
                          <!-- SKU -->
                    <value><string></string></value>
                          <!-- Description -->
                    <value><string>License 100 user Recurring fee
for</string></value>
                          <!-- Quantity -->
                    <value><double>1.000000</double></value>
                          <!-- UOM -->
                    <value><string>month(s)</string></value>
                          <!-- Duration -->
                    <value><double>1.000000</double></value>
                          <!-- Period -->
                    <value><string>month(s)</string></value>
                          <!-- UnitPrice -->
                    <value><string>USD 66.0000</string></value>
                          <!-- TaxCategory -->
                    <value><string>Default</string></value>
                          <!-- DiscountAmount -->
                    <value><string>USD 0.0000</string></value>
                          <!-- ExtendedPrice -->
                    <value><string>USD 66.0000</string></value>
                          <!-- StartDate -->
                    <value><i4>1338408000</i4></value>
                          <!-- EndDate -->
                    <value><i4>1341000000</i4></value>
                          <!-- Subscription -->
                    <value><i4>1000084</i4></value>
                          <!-- DetailType -->
                    <value><i4>110</i4></value>
                          <!-- ResourceDescription -->
                    <value><string>another plan</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

OrderPaymentsReceived_API

	Parameters	
Description	**Passed**	**Returned**
Method releases sales order payment or refund with or without specified reference number. If order payment or refund is released successfully, method calls respective order flow for sales order. If release of payment or refund is failed, method returns error code and description, order is left as it was.	2	2

Syntax

```
ItemResult BM::OrderPaymentsReceived_API(
  Int OrderID;
  Str RefNumber.
  )
returns
  Int ErrorCode - code of the occurred error;
  Str ErrorDescr - description of the error.
```

Special Notes

- **Int** OrderID - sales order ID;
- **Str** RefNumber - reference number of payment or refund attached to the order. The parameter should be passed empty if payment or refund you want to release has no reference number.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>OrderPaymentsReceived_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OrderID -->
         <value><i4>123</i4></value>
        <!-- RefNumber -->
         <value>12348567</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Error code -->
              <value><i4>0</i4></value>
              <!-- Error Description in base64
                 format (Operation Done.) -->
              <value><string>T3BlcmF0aW9uIERvbmU=</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

OrderProvisioningItemsListGet_API

	Parameters	
Description	Passed	Returned
Method retrieves list of order items bound to order.	2	19

Syntax

```
ListResult BM::OrderProvisioningItemsListGet_API(
  Int OrderID;
  Int SortNo.
  )
returns
  Int OIID - order ID;
  Int ParentOItemID - ID of parent order item to which current item subjects;
  Int OIType - type of order item (creation, upgrade, etc.);
  Int OrderDocOrderID - ID of order this order item comprises;
  Int Status - status of order item (new, provisioning, failed, etc.);
  Int StatusDesired - target status of order item;
  Str ProcessingComment - comment from provision subsystem;
  Int StartDate - start date of order item execution;
  Int EndDate - end date of order item execution;
  Int LastFDate - last provisioning order date;
  Str Descr - description;
  Int CreateTime - time of creation;
  Int ApplyTime - time of application to service;
  Int PlanID - plan ID this order item refers to;
  Int PlanPeriodID - plan period ID this order item refers to;
  Int ParentPlanPlanID - parent plan ID for plan referenced by PlanID;
  Int ParentSubscrPeriod - subscription plan period for plan referenced by PlanID;
  Int ParentSubscrPeriodType - type of subscription period referenced by
ParentSubscrPeriod;
  Int SubscriptionID - subscription ID this order item is applicable to.
```

Special Notes

- Int `OrderID - ID of order of interest;`
- Int `SortNo - Column to sort the resulting table.`

Note: method throws exception if there is no order referenced by OrderID, or if order item referenced by OrderID isn't accessible by the caller.

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>OrderProvisioningItemsListGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                                  <!-- OrderID  -->
                  <value><i4>71</i4></value>
                                  <!--  SortNo -->
                  <value><i4>0</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                        <value>
                          <array>
                            <data>
                                        <!-- OIID -->
                              <value>
                                <i4>73</i4>
                              </value>
                                        <!-- ParentOItemID -->
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                                        <!-- OIType -->
                              <value>
                                <i4>60</i4>
                              </value>
                                        <!-- OrderDocOrderID -->
                              <value>
                                <i4>71</i4>
                              </value>
                                        <!-- Status -->
                              <value>
                                <i4>40</i4>
                              </value>
                                        <!-- StatusDesired -->
                              <value>
                                <i4>10</i4>
                              </value>
                                        <!-- ProcessingComment -->
                              <value>
                                <string>
                                </string>
                              </value>
                                        <!-- StartDate -->
                              <value>
                                <i4>1334611429</i4>
                              </value>
                                        <!-- EndDate -->
                              <value>
                                <i4>1366147429</i4>
                              </value>
                                        <!-- LastFDate -->
                              <value>
                                <i4>1334606400</i4>
                              </value>
                                        <!-- Descr -->
                              <value>
                                <string>
                                        Upgrade/Downgrade Resource
&apos;Parallels Plesk Panel 10 Plus for
       Unix/Linux&apos; in Subscription #1000029.
                                      </string>
                              </value>
                                        <!-- CreateTime -->
                              <value>
                                <i4>1334611440</i4>
                              </value>
                                        <!-- ApplyTime -->
                              <value>
                                <i4>1334611491</i4>
                              </value>
                                        <!-- PlanID  -->
                              <value>
                                <i4>9</i4>
                              </value>
                                        <!-- PlanPeriodID -->
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                                        <!-- PlanVersion -->
                              <value>
                                <i4>1</i4>
                              </value>
                                        <!-- ParentPlanPlanID -->
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                                        <!-- ParentSubscrPeriod -->
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                                        <!-- ParentSubscrPeriodType -->
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                                        <!-- SubsriptionID -->
                              <value>
                                <i4>1000029</i4>
                              </value>
                            </data>
                          </array>
                        </value>
                        <value>
                          <array>
                            <data>
                              <value>
                                <i4>74</i4>
                              </value>
                              <value>
                                <i4>73</i4>
                              </value>
                              <value>
                                <i4>60</i4>
                              </value>
                              <value>
                                <i4>71</i4>
                              </value>
                              <value>
                                <i4>40</i4>
                              </value>
                              <value>
                                <i4>10</i4>
                              </value>
                              <value>
                                <string>
                                </string>
                              </value>
                              <value>
                                <i4>1334611429</i4>
                              </value>
                              <value>
                                <i4>1366147429</i4>
                              </value>
                              <value>
                                <i4>1334606400</i4>
                              </value>
                              <value>
                                <string>
                                        Upgrade / Downgrade Resource
&apos;Unlimited Domains&apos; in
      Subscription #1000029.
                                    </string>
                              </value>
                              <value>
                                <i4>1334611440</i4>
                              </value>
                              <value>
                                <i4>1334611492</i4>
                              </value>
                              <value>
                                <i4>9</i4>
                              </value>
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                              <value>
                                <i4>1</i4>
                              </value>
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                              <value>
                                <i4>-2147483648</i4>
                              </value>
                              <value>
                                <i4>1000029</i4>
                              </value>
                            </data>
                          </array>
                        </value>
                      </data>
                    </array>
                  </value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodResponse>
```

OrderProvisioningItemDetailsGet_API

Description	Parameters	
	Passed	Returned
Method retrieves details for given order item.	1	19

Syntax

```
ListResult BM::OrderProvisioningItemDetailsGet_API(
  Int OrderID.
  )
returns
  Int OIID - order ID;
  Int ParentOItemID - ID of parent order item to which current item subjects;
  Int OIType - type of order item (creation, upgrade, etc.);
  Int OrderDocOrderID - ID of order this order item comprises;
  Int Status - status of order item (new, provisioning, failed, etc.);
  Int StatusDesired - target status of order item;
  Str ProcessingComment - comment from provision subsystem;
  Int StartDate - start date of order item execution;
  Int EndDate - end date of order item execution;
  Int LastFDate - last provisioning order date;
  Str Descr - description;
  Int CreateTime - time of creation;
  Int ApplyTime - time of application to service;
  Int PlanID - plan ID this order item refers to;
  Int PlanPeriodID - plan period ID this order item refers to;
  Int ParentPlanPlanID - parent plan ID for plan referenced by PlanID;
  Int ParentSubscrPeriod - subscription plan period for plan referenced by PlanID;
  Int ParentSubscrPeriodType - type of subscription period referenced by
ParentSubscrPeriod;
  Int SubscriptionID - subscription ID this order item is applicable to.
```

Special Notes

- Int `OrderID - ID of order item of interest;`

Note: method throws exception if there is no order item referenced by OrderID, or if order item referenced by OrderID isn't accessible by the caller.

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
  <methodName>Execute</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Server</name>
            <value>BM</value>
          </member>
          <member>
            <name>Method</name>
            <value>OrderProvisioningItemDetailsGet_API</value>
          </member>
          <member>
            <name>Params</name>
            <value>
              <array>
                <data>
                                    <!-- OrderID  -->
                  <value><i4>73</i4></value>
                </data>
              </array>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>Result</name>
            <value>
              <array>
                <data>
                  <value>
                    <array>
                      <data>
                                          <!-- OIID -->
                             <value>
                           <i4>73</i4>
                       </value>
                                          <!-- ParentOItemID -->
                       <value>
```

```
                             <i4>-2147483648</i4>
                        </value>
                                  <!-- OIType -->
                        <value>
                            <i4>60</i4>
                        </value>
                                  <!-- OrderDocOrderID -->
                        <value>
                             <i4>71</i4>
                        </value>
                                  <!-- Status -->
                        <value>
                            <i4>40</i4>
                        </value>
                                  <!-- StatusDesired -->
                        <value>
                            <i4>10</i4>
                        </value>
                                  <!-- ProcessingComment -->
                        <value>
                        <string>
                        </string>
                        </value>
                                  <!-- StartDate -->
                        <value>
                            <i4>1334611429</i4>
                        </value>
                                  <!-- EndDate -->
                        <value>
                            <i4>1366147429</i4>
                        </value>
                                  <!-- LastFDate -->
                        <value>
                            <i4>1334606400</i4>
                        </value>
                                  <!-- Descr -->
                        <value>
                        <string>
                                  Upgrade/Downgrade Resource &apos;Parallels Plesk
Panel 10 Plus for                                    Unix/Linux&apos; in
Subscription #1000029.
                              </string>
                        </value>
                                  <!-- CreateTime -->
                        <value>
                            <i4>1334611440</i4>
                        </value>
                                  <!-- ApplyTime -->
                        <value>
                            <i4>1334611491</i4>
                        </value>
                                  <!-- PlanID  -->
                        <value>
                            <i4>9</i4>
                        </value>
                                  <!-- PlanPeriodID -->
                        <value>
                            <i4>-2147483648</i4>
                        </value>
                                  <!-- ParentPlanPlanID -->
                        <value>
                            <i4>-2147483648</i4>
                        </value>

                                  <!-- ParentSubscrPeriod -->
                        <value>
                            <i4>-2147483648</i4>
                        </value>

                                  <!-- ParentSubscrPeriodType -->
                        <value>
                            <i4>-2147483648</i4>
                        </value>

                                  <!-- SubsriptionID -->
                        <value>
                            <i4>1000029</i4>
                        </value>
                       </data>
                     </array>
                     </value>
                         </data>
                 </array>
                </value>
               </member>
              </struct>
            </value>
           </param>
          </params>
        </methodResponse>
```

OrderStatusChange_API

Description	Parameters	
	Passed	Returned
The method pushes specified order in the direction identified by the status.	3	1

Syntax

```
ErrorMessage BM :: OrderStatusChange_API (
Int OrderID - requested order ID,
Str NewOrderStatus - the status the order will be pushed to. The list of statuses
available depends on order type. It can be viewed under the 'Statuses' tab of the
order type selected.
Str Signature - md5 sum of the string with order data.
)
returns
Str ErrorMessage - message after specified order has its status changed: "Operation
done".
```

Special Notes

Str NewOrderStatus should be taken into account in the order flow: the desired status of the order must be specified as success status of the order flow transition.

Str Signature is md5 sum of the following string:

```
<OrderID> + <OrderNbr> + <CreationTime> + <Total> + <Descr>
```

1. `OrderID, OrderNbr, CreationTime, Total, Descr` are Order ID, Order Number, Order Creation Time, Order Total, Order Description respectively.
2. All components of the string must be concatenated; no spaces allowed between the values.
3. The values of `Order Number` and `Order Description` must be trimmed from both ends.
4. If the `Order Description` contains line breaks, they must be replaced with `\r\n`. This is important for correct signature generation.

The string can be generated for an order after getting its details by GetOrder_API (on page 310) method.

Important!
1. In spite of the fact that GetOrder_API (on page 310) method returns `order.Total` in the "11.11" format, adding the order currency prefix is required when generating the string for signature (for example, "USD 11.11" should be used instead of "11.11").
2. Keep in mind that the response can contain special symbols, such as `&amp;` instead of `&`, so you need to unescape the XML file.

Here is the example of a proper raw string for generating the md5 sum:

4290SO00000813215123 50USD 64.00Subscription on Plan #351 (SaaS.Webils Basic) for 1 Month(s).

There are only valid spaces in the example above:

- Spaces in the `Order Description` sub-string, which is normal (trimmed from ends).
- A space 'USD 64.00' sub-string, occurrence of which has been already explained.

The md5 sum for given example:

```
# echo -n "4290SO0000081321512350USD 64.00Subscription on Plan #351 (SaaS.Webils Basic) for 1 Month(s)." | md5sum

62969231047e624c35ce65d81ebaa9ac  -
```

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>OrderStatusChange_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OrderID -->
              <value><i4>47</i4></value>
              <!-- NewOrderStatus -->
              <value><string>OP</string></value>
              <!-- Signature -->
              <value><string>819f865e2b838903cc8e64cd5c9c7e65</string></value>
         </data>
        </array>
       </value>
      </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
```

```
                <value><string>
                 Operation done.
               </string></value>
              </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

PAUserPasswdUpdate_API

Method updates user password both on PBA and POA (or other external system, if a specific handler is configured for it) sides. The method signature and return parameters are identical to those ones in UserPasswdUpdate_API (on page 644).

ParentSubscriptionGet_API

	Parameters	
Description	**Passed**	**Returned**
The method returns the identifier of parent subscription for the specified subscription.	1	1 or null (if the specified subscription does not have a parent subscription)

Syntax

```
ItemResult BM::ParentSubscriptionGet_API(
  Int SubscriptionID.
)
returns
  Int ParentSubscriptionID.
```

Parameters Description

Input Parameter:

- **Int** SubscriptionID – Identifier of the subscription for which the parent subscription is to be found.

Output Parameter:

- **Int** ParentSubscriptionID – Identifier of the parent subscription for the subscription specified.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments. REMOVE all the comments from an actual request -->
    <methodCall>
      <methodName>Execute</methodName>
        <params>
          <param>
            <value>
              <struct>
                <member>
                  <name>Server</name>
                  <value>BM</value>
                </member>
                <member>
                   <name>Method</name>
                   <value>ParentSubscriptionGet_API</value>
                </member>
                <member>
                   <name>Params</name>
                   <value>
                    <array>
                      <data>
                        <!-- SubscriptionID. -->
                        <value><int>1000003</int></value>
                      </data>
                    </array>
                   </value>
                </member>
              </struct>
            </value>
          </param>
       </params>
     </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <!-- ParentSubscriptionID. -->
             <value><i4>1000001</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodResponse>
```

PaymentInfoGet_API

Description	Parameters	
	Passed	**Returned**
This method returns all the relevant information about a payment specified (*DocID* is the payment identifier).	1	27

Syntax

```
BM :: PaymentInfoGet_API (
  Int DocID.
)
returns
  Int DocID;
  Str DocNum;
  Int DocType;
  Int Status;
  Str CurrencyID;
  Double Total;
  Double TaxTotal;
  Double DocBBalance;
  Double AvailBal;
  Int Vendor_AccountID;
  Int Customer_AccountID;
  Int Crtd_DateTime;
  Int Crtd_User_UsersID;
  Int DocDate;
  Str Description;
  Str BranchID;
  Str SalesID;
  Int PayToolID;
  Int ETransStatus;
  Str FromIP;
  Str AuthCode;
  Str TrNumber;
  Int PTransActivityID;
  Int Benef_AccountID;
  Int PluginID;
  Int ChrgAttempts;
  Int PayToolType.
```

Parameters Description

Input parameter:

Int DocID - Identifier of the payment document the details for which are to be displayed.

Output parameters:

- **Int** DocID - Identifier of the payment document.
- **Str** DocNum - Number of the document.
- **Int** DocType - Type of the document. The meanings of the type value are as follows:
  - `50` - Payment
  - `70` - Refund
- **Int** Status - Status of the document. The status possible values are:
  - `1000` - Open
  - `2000` - Hold
  - `3000` - Closed
  - `4000` - Voided
  - `5000` - Refunded
  - `6000` - Cancelled
  - `7000` - Delayed
- **Str** CurrencyID - Identifier of the currency used for payment.
- **Double** Total - *Total* purchase sum specified in the document.
- **Double** TaxTotal -- *Total Tax* sum specified in the document. The value of this field is `null` because this parameter is not applicable to the *payment* and *refund* document types.
- **Double** DocBBalance - *Accounting Balance* specified in the document.
- **Double** AvailBal - *Available Balance* specified in the document.
- **Int** Vendor_AccountID - Identifier of the vendor account.
- **Int** Customer_AccountID - Identifier of the customer account.
- **Int** Crtd_DateTime - Payment creation date an time.
- **Int** Crtd_User_UsersID - Identifier of the user who has performed the payment.
- **Int** DocDate - Document issuing date.
- **Str** Description - Description of the document.
- **Str** BranchID - Identifier of the sales branch associated with the account. The value of this field is `null` because this parameter is not applicable to the *payment* and *refund* document types.
- **Str** SalesID - Identifier of the sales person associated with the account. The value of this field is `null` because this parameter is not applicable to the *payment* and *refund* document types.
- **Int** PayToolID - Identifier of the payment method used to process the payment.

- **Int** ETransStatus - Status of the payment processing. The status can be one of the following:
  - `0` - New
  - `1` - Approved by Vendor
  - `2` - Approved by Gate
  - `3` - Declined by Gate
  - `4` - Declined by Vendor
  - `5` - Fraud
  - `6` - Auth Call
  - `7` - Money Captured
  - `8` - Void
  - `9` - Refunded
  - `10` - Authorizing
  - `11` - Capturing
  - `12` - Selling
  - `13` - Voiding Auth
  - `14` - Voiding Capture
  - `15` - Voiding Sell
  - `16` - Voiding Refund
  - `17` - Refunding
  - `18` - Voiding Redirect
  - `19` - Redirecting
  - `100` - Canceled
- **Str** FromIP - IP address of the customer from which the payment is performed.
- **Str** AuthCode - Authorization code.
- **Str** TrNumber - Payment reference number.
- **Int** PTransActivityID - Identifier of the previous transaction used for refund.
- **Int** Benef_AccountID - Account that actually receives the money charged (applicable, for example, in case of reselling).
- **Int** PluginID - Identifier of the plug-in (processing center) used to process the payment.
- **Int** ChrgAttempts - Number of attempts to charge for the purchase.
- **Int** PayToolType - Type of the payment method used to process the payment. The type can be one of the following:

- `0` - Credit Card
- `1` - Bank Account
- `2` - External Redirect
- `3` - Check/Cash
- `5` - Direct Debit
- `6` - Universal

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>PaymentInfoGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- DocID-->
           <value><i4>130</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <!-- DocID -->
             <value><i4>130</i4></value>
             <!-- DocNum -->
             <value><string>000075</string></value>
             <!-- DocType -->
             <value><i4>50</i4></value>
             <!-- Status -->
             <value><i4>3000</i4></value>
             <!-- CurrencyID -->
             <value><string>USD</string></value>
             <!-- Total -->
             <value><double>50.000000</double></value>
             <!-- TaxTotal -->
             <value><double>0.000000</double></value>
             <!-- DocBBalance -->
             <value><double>0.000000</double></value>
             <!-- AvailBal -->
             <value><double>0.000000</double></value>
             <!-- Vendor AccountID -->
             <value><i4>1</i4></value>
             <!-- Customer AccountID -->
             <value><i4>1000002</i4></value>
             <!-- Crtd DateTime -->
             <value><i4>1367595785</i4></value>
             <!-- User  UsersID -->
             <value><i4>1</i4></value>
             <!-- DocDate-->
             <value><i4>1367524800</i4></value>
             <!-- Description -->
             <value><string>Payment for Order #SO000029.</string></value>
             <!-- BranchID -->
             <value><string>123</string></value>
             <!-- SalesID -->
             <value><string>456</string></value>
             <!-- PayToolID-->
             <value><i4>0</i4></value>
             <!-- ETransStatus-->
             <value><i4>0</i4></value>
             <!-- FromIP -->
             <value><string>10.6.125.124</string></value>
             <!-- AuthCode -->
             <value><string></string></value>
             <!-- TrNumber -->
             <value><string></string></value>
             <!-- PTransActivityID -->
             <value><i4>-2147483648</i4></value>
             <!-- Benef  AccountID -->
             <value><i4>1</i4></value>
             <!-- PluginID -->
             <value><i4>-2147483648</i4></value>
             <!-- ChrgAttempts -->
             <value><i4>0</i4></value>
             <!-- PayToolType -->
             <value><i4>3</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
```

```
  </params>
</methodResponse>
```

PaymentMethodGet_API

	Parameters	
Description	**Passed**	**Returned**
This method returns all the relevant information about a payment method specified (*PayToolID* is the payment method identifier).	1	5

Syntax

```
BM ::PaymentMethodGet_API(
  Int PayToolID.
)
returns
  Str PaySystem;
  Int PayToolType;
  Int OwnerAccountID;
  Int UseForAutoPayments;
  Int Status.
```

Parameters description

Input parameter:

Int PayToolID

Output parameters:

- **Str** PaySystem - Identifier of the payment system in ASCII (for example, Amex).
- **Int** PayToolType - Type of the payment method. The type can be one of the following:
  - `0` - Credit card
  - `1` - Bank account
  - `2` - External redirect
  - `3` - Check/Cash
  - `4` - External redirect for a bank account
  - `5` - Direct debit
  - `6` - Universal
- **Int** OwnerAccountID - Unique identifier of the account using the payment method registered in BM.

- **Int** UseForAutoPayments - Parameter defining whether the autopayment is used for the specified payment method or not:
  - `0x0000 0001` if *Yes*
  - `0x0000 0000` if *No*
- **Int** Status - Status of the payment method. The status can be one of the following:
  - `10` - OK
  - `20` - Fraud
  - `30` - Disabled
  - `50` - Expired
  - `60` - Suspend
  - `70` - Pending
  - `80` - Cancelling
  - `90` - Failed
  - `100` - Deleted
  - `110` - Cancelled
  - `120` - Not Activated
  - `130` - Suspended by Gate

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>PaymentMethodGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <value><string>TelstraPS</string></value>
            <value><i4>1</i4></value>
            <value><i4>1000002</i4></value>
            <value><i4>0</i4></value>
            <value><i4>10</i4></value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

PayToolListForAccountGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of available payment methods for specified account.	2	11

Syntax

```
ListResult BM::PayToolListForAccountGet_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int PayToolID - ID of the customer payment method;
  Int PayToolType - type of payment method ("0" - credit card, "1" - bank account,
"2" - external redirect, "3" - Check/Cash);
  Str PaySystemID - payment system ID;
  Str(40) PaySystem - payment system name (Visa, Mastercard and the like);
  Str(40) CutNumber - credit card number. For security reasons, it is masked and
only the last four digits are shown;
  Str(40) ExpDate - credit card expiration date (MM/YY);
  Str(40) CardHolderName - card holder name. If encryption is enabled it is
returned as "...encrypted...".;
  Str(80) AccHolderName - bank account holder name;
  Str(20) BankNumber - bank code;
  Str(24) AccountNumber - bank account number. For security reasons, it is masked
and only the last four digits are shown;
  Int UseForAutoPayments signifies whether payment method is used for automatic
payments (1 - default payment method; 2 - NOT default payment method).
```

Special Notes

- **Int** AccountID - ID of the registered in PBA account;
- **Int** SortNo - defines how output data is sorted: "1" - the output is sorted by payment method ID in ascending order, "2" - the output is sorted by payment method type in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PayToolListForAccountGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000001</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>

                <!-- PayTool ID -->
                <value><i4>0</i4></value>

                <!-- PayTool Type -->
                <value><i4>3</i4></value>

                <!-- PaySystem ID -->
                <value><string>Check/Cash</string></value>

                <!-- PaySystem -->
                <value><string>Check/Cash</string></value>

                <!-- CC Number -->
                <value><string></string></value>

                <!-- CC Expiration Date -->
                <value><string></string></value>

                <!-- CardHolder Name -->
                <value><string></string></value>

                <!-- AccHolder Name -->
                <value><string></string></value>

                <!-- Bank Number -->
                <value><string></string></value>

                <!-- Account Number -->
                <value><string></string></value>

                <!-- UseForAutoPayments -->
                <value><i4>0</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>

                <!-- PayTool ID -->
                <value><i4>7</i4></value>

                <!-- PayTool Type -->
                <value><i4>0</i4></value>

                <!-- PaySystem ID -->
                <value><string>Visa</string></value>

                <!-- PaySystem -->
                <value><string>Visa</string></value>

                <!-- CC Number -->
                <value><string>****1111</string></value>

                <!-- CC Expiration Date -->
                <value><string>05/10</string></value>
                <!-- CardHolder Name -->
                <value><string>...encrypted...</string></value>
                <!-- AccHolder Name -->
                <value><string></string></value>
                <!-- Bank Number -->
                <value><string></string></value>
                <!-- Account Number -->
                <value><string></string></value>
                <!-- UseForAutoPayments -->
                <value><i4>1</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PayToolTypeListForAccountGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of payment systems configured under specified vendor (**System** > **Settings** > **Payments Processing** submenu).	3	10

Syntax

```
ListResult BM::PayToolTypeListForAccountGet_API(
  Int VendorAccountID;
  Str CountryID;
  Int SortNo;
  )
returns
  Int PayToolID - internal ID of the payment system;
  Str PaySystem - payment system name;
  Str PaySystemName - payment system name (multi-language).
  Int PayToolType - payment system type: "0" - credit card, "1" - redirect; "2" -
direct debit; "3" - check / cash.
  Int IssueNumApplicable - signifies whether issue number for credit card is
required: "0" - not required, "1" - required. If PayToolType is not credit card,
the parameter returns "0";
  Int StartDateApplicable - signifies whether start date for credit card is
required: "0" - not required, "1" - required. If PayToolType is not credit card,
the parameter returns "0";
```

```
  Int CvvRequired - signifies whether CVV code for credit card is required: "0" -
not required, "1" - required. If PayToolType is not credit card, the parameter
returns "0";
  Int BankNumberStatus - parameter is applicable for bank accounts only. Signifies
whether to show the "Bank Number" field on payment method registration or not.
Possible values: "0" - Hidden, "1" - shown and optional, "2" - shown and
required.
  Int AccHolderNameStatus  - parameter is applicable for bank accounts only.
Signifies whether to show the "Bank Account Holder" field on payment method
registration or not. Possible values: "0" - Hidden, "1" - shown and optional, "2"
- shown and required.
  Int RegisteredPlaceStatus - parameter is applicable for bank accounts only.
Signifies whether to show the "Bank Registered Place" field on payment method
registration or not. Possible values: "0" - Hidden, "1" - shown and optional, "2"
- shown and required.
```

Special Note

- **Int** VendorAccountID - vendor account ID;
- **Str** CountryID - two-letter country code, obsolete parameter - can be passed empty;
- **Int** SortNo - defines how output data is sorted: "1" - the output is sorted by payment system ID in ascending order, "2" - the output is sorted by payment system name in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PayToolTypeListForAccountGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- CountryID -->
         <value></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                 <data>

                  <!-- PayTool ID -->
                  <value><i4>8</i4></value>

                  <!-- PaySystem -->
                  <value><string>Visa</string></value>

                  <!-- PaySystem Name-->
                  <value><string>Visa</string></value>

                  <!-- PayTool Type -->
                  <value><i4>0</i4></value>

                  <!-- IssueNumApplicable -->
                  <value><i4>0</i4></value>

                  <!-- StartDateApplicable -->
                  <value><i4>0</i4></value>

                  <!-- CVVRequired -->
                  <value><i4>0</i4></value>

                  <!-- Bank Number Status (relevant for bank accounts only) -->
                  <value><i4>2</i4></value>

                  <!-- Account Holder Name Status (relevant for bank accounts only) -->
                  <value><i4>2</i4></value>

                  <!-- Registered Place Status (relevant for bank accounts only) -->
                  <value><i4>0</i4></value>
                 </data>
                </array>
               </value>
               <value>
                <array>
                 <data>

                  <!-- PayTool ID (obsolete) -->
                  <value><i4>0</i4></value>

                  <!-- PaySystem -->
                  <value><string>Check/Cash</string></value>

                  <!-- PayTool Type -->
                  <value><i4>3</i4></value>

                  <!-- IssueNumApplicable -->
                  <value><i4>0</i4></value>

                  <!-- StartDateApplicable -->
                  <value><i4>0</i4></value>

                  <!-- CVVRequired -->
                  <value><i4>0</i4></value>

                  <!-- Bank Number Status (relevant for bank accounts only) -->
                  <value><i4>2</i4></value>
                  <!-- Account Holder Name Status (relevant for bank accounts only) -->
                  <value><i4>2</i4></value>

                  <!-- Registered Place Status (relevant for bank accounts only) -->
                  <value><i4>0</i4></value>
                 </data>
                </array>
               </value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
   </params>
  </methodResponse>
```

PayToolTypeListForAccountWithCurrencyGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of payment systems configured for actual usage under specified vendor and for specified currency (**System** > **Settings** > **Payments Processing** submenu).	4	12

Syntax

```
ListResult BM::PayToolTypeListForAccountWithCurrencyGet_API(
  Int VendorAccountID;
  Str CurrencyID
  Str CountryID;
  Int CustomerAccountID;
  Int SortNo;
  )
returns
  Int PayToolID - obsolete, returns "0" for cash, "-2" for credit cards, and for
external redirect or custom check/cash payment system it returns the Payment
Method ID value taken from a payment method settings stored in PBA;
  Str PaySystemID - payment system ID;
  Str PaySystemName - payment system name;
  Int PayToolType - payment system type: "0" - credit card, "1" - bank account; "2"
- external redirect; "3" - check / cash; "4" - external redirect for bank
account; "5" - direct debit.
  Int IssueNumApplicable - signifies whether issue number for credit card is
required: "0" - not required, "1" - required. If PayToolType is not credit card,
the parameter returns "0";
  Int StartDateApplicable - signifies whether start date for credit card is
required: "0" - not required, "1" - required. If PayToolType is not credit card,
the parameter returns "0";
  Int CvvRequired - signifies whether CVV code for credit card is required: "0" -
not required, "1" - required. If PayToolType is not credit card, the parameter
returns "0";
  Str PaperlessAgreement - paperless direct debit agreement text;
  Str DDGuarantee - direct debit refund agreement text;
  Int BankNumberStatus - signifies the Bank Number field status: "0" - the field is
hidden, "1" - the field is optional, "2" - the field is mandatory to fill;
  Int AccHolderNameStatus - signifies the Bank Account Holder field status: "0" - the
field is hidden, "1" - the field is optional, "2" - the field is mandatory to
fill;
  Int RegisteredPlaceStatus signifies the Bank Registered Place field status: "0" -
the field is hidden, "1" - the field is optional, "2" - the field is mandatory to
fill.
```

Special Note

- **Int** `VendorAccountID` - vendor account ID;

- **Str** `Currency` - three-letter currency code;
- **Str** `CountryID` - two-letter country code, obsolete parameter - can be passed empty;
- **Int** `CustomerAccountID` - customer account ID. The parameter is optional and can be passed empty. If a customer account is specified the method returns only payment systems of payment methods configured for this customer.
- **Int** `SortNo` - defines how output data is sorted: `"1"` - the output is sorted by payment system ID in ascending order, `"2"` - the output is sorted by payment system name in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PayToolTypeListForAccountWithCurrencyGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- CurrencyID -->
         <value>USD</value>
         <!-- CountryID -->
         <value></value>
         <!--CustomerAccountID -->
         <value><i4></i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                 <data>

                  <!-- PayTool ID (obsolete) -->
                  <value><i4>-2</i4></value>

                  <!-- PaySystemID -->
                  <value><string>Visa</string></value>

                  <!-- PaySystemName -->
                  <value><string>Visa</string></value>

                  <!-- PayTool Type -->
                  <value><i4>0</i4></value>

                  <!-- IssueNumApplicable -->
                  <value><i4>0</i4></value>

                  <!-- StartDateApplicable -->
                  <value><i4>0</i4></value>

                  <!-- CVVRequired -->
                  <value><i4>0</i4></value>

                  <!-- PaperlessAgreement -->
                  <value><string></string></value>

                  <!-- DDGuarantee -->
                  <value><string></string></value>

                  <!-- BankNumberStatus -->
                  <value><i4>2</i4></value>

                  <!-- AccHolderNameStatus -->
                  <value><i4>2</i4></value>

                  <!-- RegisteredPlaceStatus -->
                  <value><i4>0</i4></value>
                 </data>
                </array>
               </value>
               <value>
                <array>
                 <data>

                  <!-- PayTool ID (obsolete) -->
                  <value><i4>0</i4></value>

                  <!-- PaySystemID -->
                  <value><string>Check/Cash</string></value>

                  <!-- PaySystemName -->
                  <value><string>Check/Cash</string></value>

                  <!-- PayTool Type -->
                  <value><i4>3</i4></value>

                  <!-- IssueNumApplicable -->
                  <value><i4>0</i4></value>
                  <!-- StartDateApplicable -->
                  <value><i4>0</i4></value>

                  <!-- CVVRequired -->
                  <value><i4>0</i4></value>

                  <!-- PaperlessAgreement -->
                  <value><string></string></value>

                  <!-- DDGuarantee -->
                  <value><string></string></value>

                  <!-- BankNumberStatus -->
                  <value><i4>2</i4></value>

                  <!-- AccHolderNameStatus -->
                  <value><i4>2</i4></value>

                  <!-- RegisteredPlaceStatus -->
                  <value><i4>0</i4></value>
                 </data>
                </array>
               </value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
   </methodResponse>
```

PlaceAccountCancellationOrder_API

Description	Parameters	
	Passed	**Returned**
Method cancels all account subscriptions and account itself.	3	1

Syntax

```
ItemResult BM::PlaceAccountCancellationOrder_API(
  Int AccountID;
  Int ReasonID;
  Str Descr;
)
returns
 Str ErrorMessage - message after specified account has been cancelled:
"Cancellation Order for Account# has been placed".
```

Special Notes

- **Int** AccountID - ID of customer account to be cancelled.
- **Int** ReasonID - ID of system reason code to be stated as reason of subscription cancellation. You can find a description of reason codes in the Provider Control Panel at **System > Settings > Order Processing > Reason Codes**.

- **Str** Descr - optional comment on the reason subscription is cancelled.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlaceAccountCancellationOrder_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>100003</i4></value>
        <!-- ReasonID -->
         <value><i4>1</i4></value>
         <!-- Descr -->
         <value>Account cancellation due to customer request</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Cancellation Order for Account#100003 has been placed.
              </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlaceOrderAndAuthorize_API

	Parameters	
Description	**Passed**	**Returned**

Method functionality: • creates new account; • staff member for the account; • registers payment method; • places order for specified service plan; • places order for existing account; • creates payment for the order; • creates payment for the order with registered payment method. The main purpose of this method is to provide a single-point functionality for PBA online store.	Depends on order structure	N (depends on request)

Note:

1 'CreateAccounAndPlaceOrder_API'; 'CreateAccountAndPlaceOrder_API'; 'PlaceOrderForAccount' - methods used in previous PBA versions can still be used. Their definitions coincide with currently used method.

2 Selected API method does not have a built-in validator for provisioning parameters passed. Use the GetAndCheckServiceParameters method for supplied parameters before passing them to PlaceOrderAndAuthorize.

3 For order placement purposes, it may be necessary to pass the Terms and Conditions ID, to accept it and proceed with order placement. The additional data slot for Terms and Conditions should be added to the method call. We show how to do this in the Example 1 (on page 411) section.

Syntax

```
ItemResult BM::PlaceOrderAndAuthorize_API(
  Int VendorAccountID;
  Int ProvisioningItemsCounter;
    Str ProvisioningItem;
      Int ProvisioningDataSlotCounter;
        {
          Int ProvisioningItemID_1;
          Int ProvisioningParametersCounter;
              {
                  Str ProvisioningParameter_1;
                   ...
              Str ProvisioningParameter_N;
              }
        ...
        Int ProvisioningItemID_N;
          Int ProvisioningParametersCounter;
              {
             Str ProvisioningParameter_1;
```

```
                 ...
             Str ProvisioningParameter_N;
             }
        }
  Int ContactDataCounter;
      {
      Str ContactDataSlot_1;
      ...
      Str ContactDataSlot_N;
      }
  Int PayToolCounter;
      {
      Str PayToolSlot_1;
      ...
      Str PayToolSlot_N;
      }
  Int AdditionalContactsDataCounter;
      {
      Int AdditionalContactID_1.
          {
          Int  NumberOfParametersInContact;
          Str  ContactInfoSlot_1;
          ...
          Str  ContactInfoSlot_N;
          }
      ...
      Int AdditionalContactID_N;
          {
          Int  NumberOfParametersInContact;
          Str  ContactInfoSlot_1;
          ...
          Str  ContactInfoSlot_N.
          }
      }
)
returns
  Int AccountID - ID of newly created customer's account;
  Int OrderID - ID of newly created sales order (do not mix with order number);
  Str(64) Login - login to Control Panel for the newly created customer (if it was
not created the default           user login is returned);
  Str(40) CreationTimeStr - order creation time;
  Int DocID - null;
  Double Total - order total, without taxes and discount amounts;
  Double TaxTotal - total of taxes;
  Double DiscTotal - total of discount applied to the order;
  Double MerchTotal - order total, including taxes and discount amount;
  Str(4096) Descr - order description;
  Str(10) OrderNbr - order number;
  Str PostMethod  - method used for external redirect payment query (POST or GET);
  Str(1024) RedirectURL - URL to processing center where the order can be paid
(empty for all but external redirect payments), the URL to processing center in
case of POST or part of URL before '?' sign in case of GET;
  Int RedirectDataCounter - number of N pairs (parameter and value) to be passed to
processing center;
    Str RedirectParameterName_1;
           Str RedirectParameterValue_1;
        ...
    Str RedirectParameterName_N;
```

```
          Str RedirectParameterValue_N.
```

Special Notes:

Special notes are presented in the following subsections. Information is divided by type of method data slots: provisioning items (on page 394), provisioning parameters (on page 397), contact data (on page 400), pay tool data (on page 403) and additional contacts data (on page 406). Each section contains examples. Examples can be used to construct method calls for different purposes.

The whole method call examples are presented in the following subsections:

Example 1 - creating account with payment method and placing order on domain registration to existing hosting subscription;

Example 2 - creating account and placing order for hosting services and domain registration;

Example 3 - placing order for existing customer account on hosting services and additional resource. Customer has registered credit card.

The example of method response is presented in the **Example 3** subsection as it the same for all requests.

Provisioning Items

Special Notes:

- **Int** `VendorAccountID` – vendor account ID;
- **Int** `ProvisioningItemsCounter` - number of provisioning items in order. One order item is created for each service plan / additional resource ordered.
- **Str** `ProvisioningItem` - provisioning Item is submitted as a string of one of the following formats:
  - *Hosting Service*:

    Format:
    `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>`

    Example: `20=36=0=-1`

    Where:

    - `<PlanID>` is ID of order service plan;
    - `<PlanPeriodID>` is ID of subscription periods of the plan;
    - `<ProvisioningItemID>` is an arbitrary positive unique ID of the provisioning item inside the order;
    - `<ParentItemID>` is ID of parent provisioning item in current order. Should be "-1" for hosting service.

- *Domain Registration*:

  Format:
  `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>`

  Example: `3=5=0=-1`

  Where:

  - `<PlanID>` is ID of order service plan;
  - `<PlanPeriodID>` is ID of subscription periods of the plan;
  - `<ProvisioningItemID>` is an arbitrary positive unique ID of the provisioning item inside the order;
  - `<ParentItemID>` is ID of parent provisioning item in current order. Should be "-1" in case purchasing domain only.

- *Hosting Service + Domain Registration*:

  Format:
  `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>` - ordering hosting service, the first provisioning item.

  `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>` - ordering domain, the second provisioning item.

  Example: `20=36=0=-1` - *ProvisioningItemID*=0 (hosting service and parent subscription for domain);

  `6=4=1=0` - *ProvisioningItemID*=1 (domain to hosting service).

  Where:

  - `<PlanID>` is ID of order service plan;
  - `<PlanPeriodID>` is ID of subscription periods of the plan;
  - `<ProvisioningItemID>` is an arbitrary positive unique ID of the provisioning item inside the order;
  - `<ParentItemID>` is ID of parent provisioning item in current order. Required when domain subscription is provisioned to parent hosting subscription. Should be "-1" for hosting service. As domain is purchased to previously ordered hosting the parameter value should be number of hosting provisioning item.

- *Domain to existing subscription*:

  Format:
  `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=s<SubscriptionID>`

  Example: `20=36=0=s102356`

  Where:

  - `<PlanID>` is ID of order service plan;

    - `<PlanPeriodID>` is ID of subscription periods of the plan;
    - `<ProvisioningItemID>` is an arbitrary positive unique ID of the provisioning item inside the order;
    - `s<SubscriptionID>` is ID of existing hosting subscription, domain is purchased to.

- *Additional Resource*:

  Format:
  `<PlanID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>` - ordering hosting service, the first provisioning item.

  `<ResourceRateID>=<PlanPeriodID>=<ProvisioningItemID>=<ParentItemID>=<Amount>=RESOURCE` - ordering additional amount of resource, the parent item is mandatory.
  Where:

- `<ResourceRateID>` is ID of service plan resource rate for ordered resource;
- `<PlanPeriodID>` is ID of subscription periods of the plan;
- `<ProvisioningItemID>` is an arbitrary positive unique ID of the provisioning item inside the order;
- `<ParentItemID>` is ID of parent provisioning item in current order, for example, hosting service.
- `<Amount>` is amount of resource ordered;

Hosting Service Example:

```
<!-- VendorAccountID -->
<value><i4>1</i4></value>

<!-- ProvisioningItemsCounter -->
<value><i4>1</i4></value>

  <!-- ProvisioningItem #0. PlanID=20, PlanPeriodID=36,

   ProvisioningItemID=0, ParentItemID is not defined -->
   <value>20=36=0=-1</value>
```

Domain Registration Example:

```
<!-- VendorAccountID -->
<value><i4>1</i4></value>

<!-- ProvisioningItemsCounter -->
<value><i4>1</i4></value>

  <!-- ProvisioningItem #0. PlanID=5, PlanPeriodID=5,

   ProvisioningItemID=0, ParentItemID= not required -->
   <value>5=5=0=1</value>
```

Hosting + Domain Registration Example:

```
<!-- VendorAccountID -->
<value><i4>1</i4></value>

<!-- ProvisioningItemsCounter -->
<value><i4>2</i4></value>

  <!-- ProvisioningItem #0. PlanID=20, PlanPeriodID=36,

   ProvisioningItemID=0, ParentItemID is not defined -->
   <value>20=36=0=-1</value>
  <!-- ProvisioningItem #1. PlanID=5, PlanPeriodID=5,
   ProvisioningItemID=1, ParentItemID=0 -->
   <value>5=5=1=0</value>
```

Hosting + Resource Example:

```
<!-- VendorAccountID -->
<value><i4>1</i4></value>
<!-- ProvisioningItemsCounter -->
<value><i4>2</i4></value>
 <!-- ProvisioningItem #0. PlanID=20, PlanPeriodID=36,
  ProvisioningItemID=0, ParentItemID is not defined -->
  <value>20=36=0=-1</value>
   <!-- ProvisioningItem #1. ResourceRateID=169, PlanPeriodID=36,
  ProvisioningItemID=1, ParentItemID=0, Amount=5 -->
  <value>169=36=1=0=5=RESOURCE</value>
```

Domain Registration to Existing Hosting Subscription Example:

```
<!-- VendorAccountID -->
<value><i4>1</i4></value>
<!-- ProvisioningItemsCounter -->
<value><i4>1</i4></value>
 <!-- ProvisioningItem #0. PlanID=5, PlanPeriodID=5,
  ProvisioningItemID=0, ParentItemID=s102356 -->
  <value>5=5=0=s102356</value>
```

Provisioning Parameters

Special Notes:

- Int `ProvisioningDataSlotCounter` - number of total data slots (includes all '`<value>..</value>`' lines following after current one) in provisioning parameters section of the order lines in the section;
- Int `ProvisioningItemID_N` - ID of provisioning item #N which the following provisioning parameters are associated with.
- Int `ProvisioningParametersCounter` - number of provisioning parameters associated with current provisioning item;
- Str `ProvisioningParameter` - information required to define service parameters. For example, it is domain name in case domain subscription is ordered; parameters required by registrar additionally, like ABN/ACN for .au zones. Provisioning parameter is submitted as a string of the following format:
  `<ParameterName>=<ParameterValue>`
  Where:
  - `<ParameterName>` is required parameter defined in the service template the service plan is based on. This parameters can be pre-configured or added manually - in case order is for some manually provisioned services (like Blackberry, gifts, and the like). The following parameters are default for listed services.
    - *Domain Registration (with or without hosting)*:

`OrderOperationType` - domain operation type, applicable values are 10 - registration, 20 - renewal, 90 - transfer;

`DomainID` - domain name;

`TransferKey` - transfer key should be submitted for all operation types. For registration and renewal - submit is empty.

`AdminContactID`, `BillingContactID`, `TechContactID`, `OwnerContactID` - these four parameters can be submitted to use separate account users as administrative, billing, personal, or technical contact information. For example, to make user with ID=1 used for administrative and billing information, and user with ID 2 - for technical, the following three parameters should be passed: `<AdminContactID=1>`, `<BillingContactID=1>`, `<TechContactID=2>`. Contact information for each user should be provided in 'AdditionalContact' slot in the end of the call. Before subscribing customer to domain service plan with separate contacts, make sure that parent hosting subscription has sufficient amount of 'Users' resource, otherwise additional users can not be created in POA.

- *Hosting Service*:

  `DomainID` - domain name, it is used as hosting name. The parameter is optional, passed in case it added to service template.

- `<ParameterValue>` is provisioning parameter value;

Hosting Service Example:

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>3</i4></value>

 <!-- ProvisioningItemID #0. -->
  <value><i4>0</i4></value>

  <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
  <value><i4>1</i4></value>

  <!-- ProvisioningParameter #1. Domain name. Not defined. -->
  <value>DomainID=</value>
```

Domain Registration Example (separate account users are specified for contacts):

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>8</i4></value>

 <!-- ProvisioningItemID #0. -->
  <value><i4>0</i4></value>

  <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
  <value><i4>6</i4></value>

  <!-- ProvisioningParameter #1. Operation type - domain registration -->
  <value>OrderOperationType=10</value>

  <!-- ProvisioningParameter #2. Domain name -->
  <value>DomainID=testdomain</value>

 <!-- ProvisioningParameter #3. Transfer Key.
 Transfer key should be submitted for all operation types.
 For registration and renewal - submit is empty as below -->
  <value>TransferKey=</value>

  <!-- ProvisioningParameter #4. Admin contact ID -->
  <value>AdminContactID=0</value>

  <!-- ProvisioningParameter #5. Billing contact ID -->
  <value>BillingContactID=0</value>

  <!-- ProvisioningParameter #6. Tech contact ID -->
  <value>TechContactID=1</value>
```

Domain Registration in the .com.au zone through AusRegistry Example

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>13</i4></value>

<!-- ProvisioningItemID #0. -->
 <value><i4>0</i4></value>

 <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
 <value><i4>11</i4></value>

 <!-- ProvisioningParameter #1. Operation type - domain registration -->
 <value>OrderOperationType=10</value>

 <!-- ProvisioningParameter #2. Domain name -->
 <value>DomainID=bluewings</value>

<!-- ProvisioningParameter #3. Transfer Key.
Transfer key should be submitted for all operation types.
For registration and renewal - submit is empty as below -->
 <value>TransferKey=</value>

 <!-- ProvisioningParameter #4. Registrant name -->
 <value>R_NAME=John Smith</value>

 <!-- ProvisioningParameter #5. Registrant ID type -->
 <value>R_ID_TYPE=1</value>

 <!-- ProvisioningParameter #6. Registrant ID -->
 <value>R_ID=Blue wings</value>

 <!-- ProvisioningParameter #7. Eligibility type -->
 <value>E_TYPE=4</value>

 <!-- ProvisioningParameter #8. Eligibility name -->
 <value>E_NAME=Blue wings</value>

 <!-- ProvisioningParameter #9. Eligibility ID type -->
 <value>E_ID_TYPE=1</value>

 <!-- ProvisioningParameter #10. Eligibility ID -->
 <value>E_ID=Blue wings</value>

 <!-- ProvisioningParameter #11. Eligibility reason -->
 <value>P_REASON=1</value>
```

Hosting + Domain Registration Example (without specifying separate account users for contacts):

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>8</i4></value>

 <!-- ProvisioningItemID #0. -->
 <value><i4>0</i4></value>

 <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
 <value><i4>1</i4></value>

 <!-- ProvisioningParameter #1. Hosting subscription name - domain name. -->
 <value>DomainID=testdomain</value>

 <!-- ProvisioningItemID #1. -->
 <value><i4>1</i4></value>

 <!-- ProvisioningParametersCounter for the ProvisioningItem #1 -->
 <value><i4>3</i4></value>

 <!-- ProvisioningParameter #1. Operation type - domain registration -->
 <value>OrderOperationType=10</value>

 <!-- ProvisioningParameter #2. Domain name -->
 <value>DomainID=testdomain</value>

 <!-- ProvisioningParameter #3. Transfer Key.
```

```
Transfer key should be submitted for all operation types.
For registration and renewal - submit is empty as below -->
 <value>TransferKey=</value>
```

Hosting + Resource Example:

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>3</i4></value>

 <!-- ProvisioningItemID #0. -->
 <value><i4>0</i4></value>

 <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
 <value><i4>1</i4></value>

 <!-- ProvisioningParameter #1. Domain name. Not defined. -->
 <value>DomainID=</value>
```

Domain Registration to Existing Hosting Subscription Example (separate account users are specified for contacts):

```
<!-- ProvisioningDataSlotCounter -->
<value><i4>8</i4></value>

<!-- ProvisioningItemID #0. -->
 <value><i4>0</i4></value>

 <!-- ProvisioningParametersCounter for the ProvisioningItem #0 -->
 <value><i4>6</i4></value>

 <!-- ProvisioningParameter #1. Operation type - domain registration -->
 <value>OrderOperationType=10</value>

 <!-- ProvisioningParameter #2. Domain name -->
 <value>DomainID=testdomain</value>

 <!-- ProvisioningParameter #3. Transfer Key.
Transfer key should be submitted for all operation types.
For registration and renewal - submit is empty as below -->
 <value>TransferKey=</value>

 <!-- ProvisioningParameter #4. Admin contact ID -->
 <value>AdminContactID=1</value>

 <!-- ProvisioningParameter #5. Billing contact ID -->
 <value>BillingContactID=1</value>

 <!-- ProvisioningParameter #6. Tech contact ID -->
 <value>TechContactID=2</value>
```

Contact Data

Special Notes:

- **Int** `ContactDataCounter` - number of contact data slots;
- **Str** `ContactDataSlot_N` - data slot #N of contact information on customer placing the order. Identifies customer account on behalf of which the order is placed. Contact data are submitted as strings of the following format:

  `<ContactDataSlotName>=<ContactDataSlotValue>`
  Where:

  - `<ContactDataSlotName>` is a name of the contact data slot;
  - `<ContactDataSlotValue>` is a value of the contact data slot;

  Parallels Business Automation supports the following contact data slots:

- *New Customer*:
  - `LoginID` - login to CP;
  - `PasswordID` - password to CP;
  - `FullyRegistred` - the parameter that indicates whether a customer completed full registration, possible values are 0 (no) and 1 (yes);
  - `CompanyNameID` - company name for company account, or first name + last name for personal accounts; (leaving this field empty will create a personal account);
  - `CompanyNameLatinID` - same as 'CompanyNameID' in ASCII charset (for international use);
  - `PostAddressID` - free-form postal address;
  - `TaxRegID` - customer's tax registration ID;
  - `FirstNameID` - customer's first name;
  - `MiddleNameID` - customer's middle name;
  - `LastNameID` - customer's last name;
  - `AddressID` - address line 1;
  - `Address2ID` - address line 2;
  - `CityID` - customer's city;
  - `ZipID` - customer's ZIP or postal code;
  - `CountryID` - customer's country;
  - `StateID` - customer's state or province;
  - `EmailID` - customer's e-mail address;
  - `PhoneCountryID` - customer's phone country code;
  - `PhoneAreaID` - customer's phone area code;
  - `PhoneNumberID` - customer's phone number
  - `PhoneExtensionID` - customer's phone number extension;
  - `FaxCountryID` - customer's fax country code;
  - `FaxAreaID` - customer's fax area code;
  - `FaxNumberID` - customer's fax number;
  - `FaxExtensionID` - customer's fax number extension;
  - `BannerID` - ID of campaign if customer has been redirected to the store by one of banner campaigns configured in the PBA;
  - `PromoCodeID` - promotion code if customer enters one;

    - `SalesPersonID` - sales person order ID assigned to account;
    - `SalesBranchID` - sales branch ID assigned to account.

Note: each additional attribute you create becomes contact data slot and can be passed among others.

- *Existing Customer*:
    - `AccountID` - ID of customer existing account;
    - `PromoCodeID` - promotion code if customer enters one.

New Customer Example:

```
<!-- ContactDataCounter -->
<value><i4>24</i4></value>

 <!-- Login to CP -->
 <value>LoginID=test123</value>

 <!-- Password to CP -->
 <value>XXXPasswordID=123test</value>

 <!-- Customer's description -->
 <value>CompanyNameID=John Smith</value>

 <!-- First Name -->
 <value>FirstNameID=John</value>

 <!-- Middle Name -->
 <value>MiddleNameID=Shawn</value>

 <!-- Last Name -->
 <value>LastNameID=Smith</value>

 <!-- Address (line 1/2) -->
 <value>AddressID=Sunrise Valley Drive</value>

 <!-- Address (line 2/2) -->
 <value>Address2ID=Suite 600</value>

 <!-- City -->
 <value>CityID=New York</value>

 <!-- State -->
 <value>StateID=NY</value>

 <!-- Zip code -->
 <value>ZipID=12345</value>

 <!-- Country -->
 <value>CountryID=us</value>

 <!-- Email -->
 <value>EmailID=jsmith@tailor.com</value>

 <!-- Phone country code -->
 <value>PhoneCountryID=1</value>

 <!-- Phone area code -->
 <value>PhoneAreaID=201</value>

 <!-- Phone number -->
 <value>PhoneNumberID=4568523</value>

 <!-- Phone Extension -->
 <value>PhoneExtensionID=245</value>

 <!-- Fax country code -->
 <value>FaxCountryID=1</value>

 <!-- Fax area code -->
 <value>FaxAreaID=201</value>

 <!-- Fax number -->
 <value>FaxNumberID=4568523</value>

 <!-- Fax Extension -->
```

```
<value>FaxExtensionID=235</value>
<!-- Promo code -->
<value>PromoCodeID=promo895</value>
<!-- Sales Person ID-->
<value>SalesPersonID=1102</value>
<!-- Sales Branch ID-->
<value>SalesBranchID=1</value>
```

Existing Customer Example:

```
<!-- ContactDataCounter -->
<value><i4>2</i4></value>
<!-- Promo code -->
<value>PromoCodeID=promo895</value>
<!-- Customer's Account ID -->
<value>AccountID=1000002</value>
```

Pay Tool Data

Special Notes:

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the CardNumberID argument must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42).

- **Int** PayToolCounter - number of payment tool slots;
- **Str** `PayToolSlot_N` – data slot #N of information about payment tool which is going to be used to pay the order. Defines payment method used to pay the order. Generally, for credit card payments it is enough to submit the `PayToolID` and arbitrary the `IPAddressID`.

  Pay Tool data are submitted as strings of the following format:

  `<PayToolSlotName>=<PayToolSlotValue>`

  Where:

  - `<PayToolSlotName>` is a name of the payment tool data slot;
  - `<PayToolSlotValue>` is a value of the payment tool data slot;

  Parallels Business Automation supports the following payment tool data slots:

  - *Check/Cash*:
    - `PayToolTypeID=3;`
    - `PluginID=0;`
    - `IPAddressID` - customers current IP address.
  - *Registered credit card*:
    - `PayToolID` - ID of registered payment method.
    - `IPAddressID` - customers current IP address.
  - *New Credit Card*:

  - `PayToolTypeID=0`; "1" - bank account, "2" - external redirect, "3" - Check/Cash;
  - `CardTypeID` - ID of payment system registered in PBA, for example. "Visa", "MasterCard", etc;
  - `CardNumberID` - CC number;

(**Note**: if credit card number is not passed with 'XXX' prefix the response will contain unencrypted CC number after calling the method);

  - `CardHolderNameID` - card holder name;
  - `CVCID` - CC security code;
  - `IssueNumID` - credit card issue number, can be skipped if payment system doesn't require;
  - `StartDateID` - credit card start date, can be skipped if payment system doesn't require;
  - `ExpDateID` - CC expiration date in format MM/YY;
  - `BillingAddressID` - first line of billing address;
  - `BillingAddress2ID` - second line of billing address;
  - `BillingCityID` - card holder's city;
  - `BillingStateID` - card holder's state;
  - `BillingZipID` - card holder's ZIP or postal code;
  - `BillingCountryID` - card holder's country;
  - `BillingEmailID` - card holder email;
  - `BillingPhoneCountryID` - phone code of card holder country;
  - `BillingPhoneAreaID` - phone area code;
  - `BillingPhoneNumberID` - phone number;
  - `BillingPhoneExtensionID` - phone extension, optional;
  - `BillingFaxCountryID` - fax code of cardholder country;
  - `BillingFaxAreaID` - fax phone area code;
  - `BillingFaxNumberID` - fax number;
  - `BillingFaxExtensionID` - fax extension, optional;
  - `IPAddressID` - customers current IP address.
- *Bank Account*:
  - `PayToolTypeID=1`;
  - `BankNumberID` - customer bank number;

- `AccountNumberID` - customer bank account number;
- `AccountHolderNameID` - account holder name;
- `IPAddressID` - customers current IP address.

- *External Redirect*:
  - `PayToolTypeID=2`;
  - `PluginID` - ID of a payment plug-in of the external redirect type, for example PayPal;
  - `PayToolID` - ID of registered payment method;
  - `IPAddressID` - customers current IP address.

Check/Cash Example:

```
<!-- PayToolCounter -->
<value><i4>3</i4></value>
 <!-- Type of Payment tool. 3 means Check/Cash -->
 <value>PayToolTypeID=3</value>
 <!-- Identifier of payment plug-in Check/Cash -->
 <value>PluginID=0</value>
 <!-- IP Address -->
 <value>IPAddressID=192.168.232.10</value>
```

Registered Credit Card Example:

```
<!-- PayToolCounter -->
<value><i4>2</i4></value>
 <!-- Identifier of Payment tool registered
on Customer's Account -->
 <value>PayToolID=3</value>
 <!-- IP Address -->
 <value>IPAddressID=192.168.232.10</value>
```

New Credit Card Example:

```
<!-- PayToolCounter -->
<value><i4>7</i4></value>
 <!-- Type of Payment tool. 0 means Credit Card -->
 <value>PayToolTypeID=0</value>
 <!-- Type of CC -->
 <value>CardTypeID=Visa</value>
<!-- Card Number -->
 <value>XXXCardNumberID=4111111111111111</value>
<!-- Cardholder name -->
 <value>CardHolderNameID=JOHN SMITH</value>
<!-- Security code -->
 <value>CVCID=123</value>
<!-- Expiration date -->
 <value>ExpDateID=07/08</value>
<!-- IP Address -->
 <value>IPAddressID=192.168.232.10</value>
```

Bank Account:

```
<!-- PayToolCounter -->
<value><i4>5</i4></value>
```

```
<!-- Type of Payment tool. 1 means Bank Account -->
 <value>PayToolTypeID=1</value>
<!-- Number of customer bank -->
 <value>BankNumberID=45689</value>
<!-- Account Number -->
 <value>AccountNumberID=356892145</value>
<!-- Accountholder name -->
 <value>AccountHolderNameID=JOHN SMITH</value>
<!-- IP Address -->
 <value>IPAddressID=192.168.232.10</value>
```

External Redirect:

```
<!-- PayToolCounter -->
<value><i4>4</i4></value>
<!-- Type of Payment tool. 2 means External Redirect -->
 <value>PayToolTypeID=2</value>
<!-- Identifier of an external redirect payment plug-in -->
 <value>PluginID=3</value>
 <!-- Identifier of Payment tool registered
on Customer's Account -->
 <value>PayToolID=3</value>
<!-- IP Address -->
 <value>IPAddressID=192.168.232.10</value>
```

Additional Contacts Data

Special Notes:

Attention! The slot is passed only in case of domain registration and one or all `AdminContactID`, `BillingContactID`, `TechContactID` and `OwnerContactID` parameters are submitted in `ProvisioningParametersSlot`.

- **Int** `AdditionalContactsDataCounter` - number of additional data slots;
- **Int** `AdditionalContactID_N` - ID of additional contact #N which the following contact information is associated with. The value should coincide with one passed in ProvisioningParametersSlot (on page 397);
- **Int** `NumberOfParametersInContact` - number of parameters associated with current additional contact.
- **Str** `ContactInfoSlot_N` - data slot #N of contact information on additional user to be used as separate domain owner information: administrative, billing, personal, and technical. Contact data are submitted as strings of the following format:

  Additional contact information is submitted in the same format as account information:

  `<AdditionalDataSlotName>=<AdditionalDataSlotValue>`

  Where:

  - `<AdditionalDataSlotName>` is a name of the additional contact data slot;
  - `<AdditionalDataSlotValue>` is a number of the additional contact data slots;

  Default user information data set has 18 parameters (+ user attributes - if any):

- `FirstNameID` - user first name;
- `MiddleNameID` - user middle name;
- `LastNameID` - user last name;
- `AddressID` - address line 1;
- `Address2ID` - address line 2;
- `CityID` - user city;
- `ZipID` - user ZIP or postal code;
- `CountryID` - user country;
- `StateID` - user state or province;
- `EmailID` - user e-mail address;
- `PhoneCountryID` - user phone country code;
- `PhoneAreaID` - user phone area code;
- `PhoneNumberID` - user phone number;
- `PhoneExtensionID` - user phone number extension;
- `FaxCountryID` - user fax country code;
- `FaxAreaID` - user fax area code;
- `FaxNumberID` - user fax number;
- `FaxExtensionID` - user fax number extension.

One Contact Example:

```
<!-- AdditionalContactDataSlotCounter --!>
<value><i4>1</i4></value>
 <!-- AdditionalContactID. For example, Admin contact --!>
 <value><i4>0</i4></value>
 <!-- NumberOfParamsInContact --!>
 <value><i4>18</i4></value>
  <!-- FName -->
   <value>FirstNameID=John</value>
  <!-- MName -->
   <value>MiddleNameID=F</value>
  <!-- LName -->
   <value>LastNameID=Smith</value>
  <!-- Address (line 1/2) -->
   <value>AddressID=Sunrise Valley Drive</value>
  <!-- Address (line 2/2) -->
   <value>Address2ID=Suite 600</value>
  <!-- City -->
   <value>CityID=New York</value>
  <!-- State -->
   <value>StateID=NY</value>
  <!-- Zip code -->
   <value>ZipID=12345</value>
  <!-- Country -->
   <value>CountryID=us</value>
  <!-- Email -->
   <value>EmailID=jsmith@tailor.com</value>
  <!-- Phone country code -->
   <value>PhoneCountryID=1</value>
  <!-- Phone area code -->
   <value>PhoneAreaID=201</value>
  <!-- Phone number -->
   <value>PhoneNumberID=4568523</value>
  <!-- Phone Extension -->
   <value>PhoneExtensionID=245</value>
   <!-- Fax country code -->
   <value>FaxCountryID=1</value>
  <!-- Fax area code -->
   <value>FaxAreaID=201</value>
  <!-- Fax number -->
   <value>FaxNumberID=4568523</value>
  <!-- Fax Extension -->
   <value>FaxExtensionID=235</value>
```

Four Contacts Example:

```
<!-- AdditionalContactDataSlotCounter --!>
<value><i4>4</i4></value>
 <!-- AdditionalContactID Admin contact--!>
 <value><i4>0</i4></value>
 <!-- NumberOfParamsInContact --!>
 <value><i4>18</i4></value>
 <!-- FName -->
  <value>FirstNameID=John</value>
 <!-- MName -->
  <value>MiddleNameID=F</value>
 <!-- LName -->
  <value>LastNameID=Smith</value>
 <!-- Address (line 1/2) -->
  <value>AddressID=Sunrise Valley Drive</value>
 <!-- Address (line 2/2) -->
  <value>Address2ID=Suite 600</value>
 <!-- City -->
  <value>CityID=New York</value>
 <!-- State -->
  <value>StateID=NY</value>
 <!-- Zip code -->
  <value>ZipID=12345</value>
 <!-- Country -->
  <value>CountryID=us</value>
 <!-- Email -->
  <value>EmailID=jsmith@tailor.com</value>
 <!-- Phone country code -->
  <value>PhoneCountryID=1</value>
 <!-- Phone area code -->
  <value>PhoneAreaID=201</value>
 <!-- Phone number -->
  <value>PhoneNumberID=4568523</value>
 <!-- Phone Extension -->
  <value>PhoneExtensionID=245</value>
 <!-- Fax country code -->
  <value>FaxCountryID=1</value>
 <!-- Fax area code -->
```

```
  <value>FaxAreaID=201</value>
 <!-- Fax number -->
  <value>FaxNumberID=4568523</value>
 <!-- Fax Extension -->
  <value>FaxExtensionID=235</value>
<!-- AdditionalContactID. Billing contact --!>
 <value><i4>1</i4></value>
<!-- NumberOfParamsInContact --!>
 <value><i4>18</i4></value>
 <!-- FName -->
  <value>FirstNameID=Mark</value>
 <!-- MName -->
  <value>MiddleNameID=J</value>
 <!-- LName -->
  <value>LastNameID=Short</value>
 <!-- Address (line 1/2) -->
  <value>AddressID=Sunrise Valley Drive</value>
 <!-- Address (line 2/2) -->
  <value>Address2ID=Suite 600</value>
 <!-- City -->
  <value>CityID=New York</value>
 <!-- State -->
  <value>StateID=NY</value>
 <!-- Zip code -->
  <value>ZipID=12345</value>
 <!-- Country -->
  <value>CountryID=us</value>
 <!-- Email -->
  <value>EmailID=mshort@tailor.com</value>
  <!-- Phone country code -->
  <value>PhoneCountryID=1</value>
 <!-- Phone area code -->
  <value>PhoneAreaID=201</value>
 <!-- Phone number -->
  <value>PhoneNumberID=4568523</value>
 <!-- Phone Extension -->
  <value>PhoneExtensionID=241</value>
 <!-- Fax country code -->
  <value>FaxCountryID=1</value>
 <!-- Fax area code -->
  <value>FaxAreaID=201</value>
 <!-- Fax number -->
  <value>FaxNumberID=4568523</value>
 <!-- Fax Extension -->
  <value>FaxExtensionID=239</value>
 <!-- AdditionalContactID Owner contact--!>
  <value><i4>2</i4></value>
 <!-- NumberOfParamsInContact --!>
  <value><i4>18</i4></value>
 <!-- FName -->
  <value>FirstNameID=Fred</value>
 <!-- MName -->
  <value>MiddleNameID=R</value>
 <!-- LName -->
  <value>LastNameID=Bonfort</value>
 <!-- Address (line 1/2) -->
  <value>AddressID=Sunrise Valley Drive</value>
 <!-- Address (line 2/2) -->
```

```
  <value>Address2ID=Suite 600</value>
<!-- City -->
  <value>CityID=New York</value>
<!-- State -->
  <value>StateID=NY</value>
<!-- Zip code -->
  <value>ZipID=12345</value>
<!-- Country -->
  <value>CountryID=us</value>
<!-- Email -->
  <value>EmailID=fbonfort@tailor.com</value>
<!-- Phone country code -->
  <value>PhoneCountryID=1</value>
<!-- Phone area code -->
  <value>PhoneAreaID=201</value>
<!-- Phone number -->
  <value>PhoneNumberID=4568523</value>
<!-- Phone Extension -->
  <value>PhoneExtensionID=243</value>
<!-- Fax country code -->
  <value>FaxCountryID=1</value>
<!-- Fax area code -->
  <value>FaxAreaID=201</value>
<!-- Fax number -->
  <value>FaxNumberID=4568523</value>
<!-- Fax Extension -->
  <value>FaxExtensionID=232</value>
<!-- AdditionalContactID. Tech contact --!>
  <value><i4>3</i4></value>
<!-- NumberOfParamsInContact --!>
  <value><i4>18</i4></value>
<!-- FName -->
  <value>FirstNameID=Jhoe</value>
<!-- MName -->
  <value>MiddleNameID=J</value>
<!-- LName -->
  <value>LastNameID=Doe</value>
<!-- Address (line 1/2) -->
  <value>AddressID=Sunrise Valley Drive</value>
<!-- Address (line 2/2) -->
  <value>Address2ID=Suite 600</value>
<!-- City -->
  <value>CityID=New York</value>
<!-- State -->
  <value>StateID=NY</value>
<!-- Zip code -->
  <value>ZipID=12345</value>
<!-- Country -->
  <value>CountryID=us</value>
<!-- Email -->
  <value>EmailID=jdoe@tailor.com</value>
<!-- Phone country code -->
  <value>PhoneCountryID=1</value>
<!-- Phone area code -->
  <value>PhoneAreaID=201</value>
<!-- Phone number -->
  <value>PhoneNumberID=4568523</value>
<!-- Phone Extension -->
```

```
  <value>PhoneExtensionID=247</value>
 <!-- Fax country code -->
  <value>FaxCountryID=1</value>
 <!-- Fax area code -->
  <value>FaxAreaID=201</value>
 <!-- Fax number -->
  <value>FaxNumberID=4568523</value>
 <!-- Fax Extension -->
  <value>FaxExtensionID=236</value>
```

Example 1

Example below creates account with payment method and places order on domain registration to existing hosting subscription. During order placement, Terms and Conditions are accepted.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlaceOrderAndAuthorize_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- ProvisioningItemsCounter -->
         <value><i4>1</i4></value>
         <!-- ProvisioningItem #0. PlanID=5, PlanPeriodID=5,
           ProvisioningItemID=0, ParentItemID=s102356 -->
         <value>5=5=0=s102356</value>
         <!-- ProvisioningDataSlotCounter -->
         <value><i4>8</i4></value>
         <!-- ProvisioningItemID #0. -->
         <value><i4>0</i4></value>
         <!-- ProvisioningParametersCounter for
            the ProvisioningItem #0 -->
         <value><i4>6</i4></value>
         <!-- ProvisioningParameter #1.
           Operation type - domain registration -->
         <value>OrderOperationType=10</value>
         <!-- ProvisioningParameter #2. Domain name -->
```

```
<value>DomainID=testdomain</value>
<!-- ProvisioningParameter #3. Transfer Key.
     Transfer key should be submitted for all operation types.
     For registration and renewal - submit is empty as below -->
<value>TransferKey=</value>
<!-- ProvisioningParameter #4. Admin contact ID -->
<value>AdminContactID=0</value>
<!-- ProvisioningParameter #5. Billing contact ID -->
<value>BillingContactID=0</value>
<!-- ProvisioningParameter #6. Tech contact ID -->
<value>TechContactID=1</value>
<!-- ContactDataCounter -->
<value><i4>22</i4></value>
<!-- Login to CP -->
<value>LoginID=test123</value>
<!-- Password to CP -->
<value>XXXPasswordID=123test</value>
<!-- Customer's description -->
<value>CompanyNameID=John "tailor" Smith</value>
<!-- First Name -->
<value>FirstNameID=John</value>
<!-- Middle Name -->
<value>MiddleNameID=Shawn</value>
<!-- Last Name -->
<value>LastNameID=Smith</value>
<!-- Address (line 1/2) -->
<value>AddressID=Sunrise Valley Drive</value>
<!-- Address (line 2/2) -->
<value>Address2ID=Suite 600</value>
<!-- City -->
<value>CityID=New York</value>
<!-- State -->
<value>StateID=NY</value>
<!-- Zip code -->
<value>ZipID=12345</value>
<!-- Country -->
<value>CountryID=us</value>
<!-- Email -->
<value>EmailID=jsmith@tailor.com</value>
<!-- Phone country code -->
<value>PhoneCountryID=1</value>
<!-- Phone area code -->
<value>PhoneAreaID=201</value>
<!-- Phone number -->
<value>PhoneNumberID=4568523</value>
<!-- Phone Extension -->
<value>PhoneExtensionID=245</value>
<!-- Fax country code -->
<value>FaxCountryID=1</value>
<!-- Fax area code -->
<value>FaxAreaID=201</value>
<!-- Fax number -->
<value>FaxNumberID=4568523</value>
<!-- Fax Extension -->
<value>FaxExtensionID=235</value>
<!-- Promo code -->
<value>PromoCodeID=promo895</value>
<!-- PayToolCounter -->
```

```
<value><i4>7</i4></value>
<!-- Type of Payment tool. 0 means Credit Card -->
<value>PayToolTypeID=0</value>
<!-- Type of CC -->
<value>CardTypeID=Visa</value>
<!-- Card Number -->
<value>CardNumberID=4111111111111111</value>
<!-- Cardholder name -->
<value>CardHolderNameID=JOHN SMITH</value>
<!-- Security code -->
<value>CVCID=123</value>
<!-- Expiration date -->
<value>ExpDateID=07/08</value>
<!-- IP Address -->
<value>IPAddressID=192.168.232.10</value>
<!-- AdditionalContactDataSlotCounter -->
<value><i4>2</i4></value>
<!-- AdditionalContactID -->
<value><i4>0</i4></value>
<!-- NumberOfParamsInContact -->
<value><i4>18</i4></value>
<!-- FName -->
<value>FirstNameID=John</value>
<!-- MName -->
<value>MiddleNameID=F</value>
<!-- LName -->
<value>LastNameID=Smith</value>
<!-- Address (line 1/2) -->
<value>AddressID=Sunrise Valley Drive</value>
<!-- Address (line 2/2) -->
<value>Address2ID=Suite 600</value>
<!-- City -->
<value>CityID=New York</value>
<!-- State -->
<value>StateID=NY</value>
<!-- Zip code -->
<value>ZipID=12345</value>
<!-- Country -->
<value>CountryID=us</value>
<!-- Email -->
<value>EmailID=jsmith@tailor.com</value>
<!-- Phone country code -->
<value>PhoneCountryID=1</value>
<!-- Phone area code -->
<value>PhoneAreaID=201</value>
<!-- Phone number -->
<value>PhoneNumberID=4568523</value>
<!-- Phone Extension -->
<value>PhoneExtensionID=245</value>
<!-- Fax country code -->
<value>FaxCountryID=1</value>
<!-- Fax area code -->
<value>FaxAreaID=201</value>
<!-- Fax number -->
<value>FaxNumberID=4568523</value>
<!-- Fax Extension -->
<value>FaxExtensionID=235</value>
<!-- AdditionalContactID. Tech contact -->
```

```
         <value><i4>1</i4></value>
         <!-- NumberOfParamsInContact -->
         <value><i4>18</i4></value>
         <!-- FName -->
         <value>FirstNameID=Mark</value>
         <!-- MName -->
         <value>MiddleNameID=J</value>
         <!-- LName -->
         <value>LastNameID=Short</value>
         <!-- Address (line 1/2) -->
         <value>AddressID=Sunrise Valley Drive</value>
         <!-- Address (line 2/2) -->
         <value>Address2ID=Suite 600</value>
         <!-- City -->
         <value>CityID=New York</value>
         <!-- State -->
         <value>StateID=NY</value>
         <!-- Zip code -->
         <value>ZipID=12345</value>
         <!-- Country -->
         <value>CountryID=us</value>
         <!-- Email -->
         <value>EmailID=mshort@tailor.com</value>
         <!-- Phone country code -->
         <value>PhoneCountryID=1</value>
         <!-- Phone area code -->
         <value>PhoneAreaID=201</value>
         <!-- Phone number -->
         <value>PhoneNumberID=4568523</value>
         <!-- Phone Extension -->
         <value>PhoneExtensionID=246</value>
         <!-- Fax country code -->
         <value>FaxCountryID=1</value>
         <!-- Fax area code -->
         <value>FaxAreaID=201</value>
         <!-- Fax number -->
         <value>FaxNumberID=4568523</value>
         <!-- Fax Extension -->
         <value>FaxExtensionID=235</value>
         <!-- NumberOfParamsIn Terms and Conditions data slot -->
         <value>1</value>
         <!-- Terms and Conditions ID -->
         <value>2</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Example 2

Example below creates account with payment method and places order on hosting services and domain registration.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlaceOrderAndAuthorize_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- ProvisioningItemsCounter -->
         <value><i4>2</i4></value>
         <!-- ProvisioningItem #0. PlanID=14, PlanPeriodID=14,
            ProvisioningItemID=0, ParentItemID is not defined -->
         <value>14=14=0=-1</value>
         <!-- ProvisioningItem #1. PlanID=2, PlanPeriodID=2,
            ProvisioningItemID=1, ParentItemID=0 -->
         <value>2=2=1=0</value>
         <!-- ProvisioningDataSlotCounter -->
         <value><i4>11</i4></value>
         <!-- ProvisioningItemID #0. -->
         <value><i4>0</i4></value>
         <!-- ProvisioningParametersCounter for
            the ProvisioningItem #0 -->
         <value><i4>4</i4></value>
         <!-- ProvisioningParameter #1. Domain name. Not defined. -->
         <value>DomainID=</value>
         <!-- ProvisioningParameter #2. Login to hosting CP. -->
         <value>LoginID=testlogin</value>
         <!-- ProvisioningParameter #3. Password to hosting CP. -->
         <value>XXXPasswordID=testpassword</value>
         <!-- ProvisioningParameter #4. Password confirmation -->
         <value>Password2ID=testpassword</value>
         <!-- ProvisioningItemID #1. -->
         <value><i4>1</i4></value>
         <!-- ProvisioningParametersCounter for
    the ProvisioningItem #1 -->
<value><i4>3</i4></value>
<!-- ProvisioningParameter #1.
       Operation type - domain registration -->
<value>OrderOperationType=10</value>
<!-- ProvisioningParameter #2. Domain name -->
<value>DomainID=testdomain</value>
<!-- ProvisioningParameter #3. Transfer Key.
       Transfer key should be submitted for all operation types.
       For registration and renewal - submit is empty as below -->
<value>TransferKey=</value>
<!-- ContactDataCounter -->
<value><i4>22</i4></value>
<!-- Login to CP -->
<value>LoginID=testlogin</value>
<!-- Password to CP -->
<value>XXXPasswordID=testpassword</value>
<!-- Customer's description -->
<value>CompanyNameID=John "tailor" Smith</value>
<!-- First Name -->
<value>FirstNameID=John</value>
<!-- Middle Name -->
<value>MiddleNameID=Shawn</value>
<!-- Last Name -->
<value>LastNameID=Smith</value>
<!-- Address (line 1/2) -->
<value>AddressID=Sunrise Valley Drive</value>
<!-- Address (line 2/2) -->
<value>Address2ID=Suite 600</value>
<!-- City -->
<value>CityID=New York</value>
<!-- State -->
<value>StateID=NY</value>
<!-- Zip code -->
<value>ZipID=12345</value>
<!-- Country -->
<value>CountryID=us</value>
<!-- Email -->
<value>EmailID=jsmith@tailor.com</value>
<!-- Phone country code -->
<value>PhoneCountryID=1</value>
<!-- Phone area code -->
<value>PhoneAreaID=201</value>
<!-- Phone number -->
<value>PhoneNumberID=4568523</value>
<!-- Phone Extension -->
<value>PhoneExtensionID=245</value>
<!-- Fax country code -->
<value>FaxCountryID=1</value>
<!-- Fax area code -->
<value>FaxAreaID=201</value>
<!-- Fax number -->
<value>FaxNumberID=4568523</value>
<!-- Fax Extension -->
<value>FaxExtensionID=235</value>
<!-- Promo code -->
<value>PromoCodeID=promo895</value>
<!-- PayToolCounter -->
```

```
         <value><i4>7</i4></value>

         <!-- Type of Payment tool. 0 means Credit Card -->
         <value>PayToolTypeID=0</value>

         <!-- Type of CC -->
         <value>CardTypeID=Visa</value>

         <!-- Card Number -->
         <value>CardNumberID=4111111111111111</value>

         <!-- Cardholder name -->
         <value>CardHolderNameID=JOHN SMITH</value>

         <!-- Security code -->
         <value>CVCID=123</value>

         <!-- Expiration date -->
         <value>ExpDateID=07/08</value>

         <!-- IP Address -->
         <value>IPAddressID=192.168.232.10</value>

         <!-- AdditionalContactDataSlotCounter -->
         <value><i4>2</i4></value>

         <!-- AdditionalContactID -->
         <value><i4>0</i4></value>

         <!-- NumberOfParamsInContact -->
         <value><i4>18</i4></value>

         <!-- FName -->
         <value>FirstNameID=John</value>

         <!-- MName -->
         <value>MiddleNameID=Shawn</value>

         <!-- LName -->
         <value>LastNameID=Smith</value>

         <!-- Address (line 1/2) -->
         <value>AddressID=Sunrise Valley Drive</value>

         <!-- Address (line 2/2) -->
         <value>Address2ID=Suite 600</value>

         <!-- City -->
         <value>CityID=New York</value>

         <!-- State -->
         <value>StateID=NY</value>

         <!-- Zip code -->
         <value>ZipID=12345</value>

         <!-- Country -->
         <value>CountryID=us</value>

         <!-- Email -->
         <value>EmailID=jsmith@tailor.com</value>

         <!-- Phone country code -->
         <value>PhoneCountryID=1</value>

         <!-- Phone area code -->
         <value>PhoneAreaID=201</value>

         <!-- Phone number -->
         <value>PhoneNumberID=4568523</value>

         <!-- Phone Extension -->
         <value>PhoneExtensionID=245</value>

         <!-- Fax country code -->
         <value>FaxCountryID=1</value>

         <!-- Fax area code -->
         <value>FaxAreaID=201</value>

         <!-- Fax number -->
         <value>FaxNumberID=4568523</value>

         <!-- Fax Extension -->
         <value>FaxExtensionID=235</value>
        </data>
```

```
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Example 3

Example below places order for customer account (ID=1000002) on hosting services and additional resource. Customer has registered credit card (payment tool) with ID=3.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlaceOrderAndAuthorize_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- VendorAccountID -->
         <value><i4>1</i4></value>
         <!-- ProvisioningItemsCounter -->
         <value><i4>2</i4></value>
         <!-- ProvisioningItem #0. PlanID=1, PlanPeriodID=3,
             ProvisioningItemID=0, ParentItemID is not defined -->
         <value>1=3=0=-1</value>
        <!-- ProvisioningItem #1. ResourceRateID=12, PlanPeriodID=3,
             ProvisioningItemID=1, ParentItemID=0, Amount=100 -->
         <value>12=3=1=0=100=RESOURCE</value>
         <!-- ProvisioningDataSlotCounter -->
         <value><i4>3</i4></value>
         <!-- ProvisioningItemID #0 -->
         <value><i4>0</i4></value>
         <!-- ProvisioningParametersCounter for
              the ProvisioningItem #0 -->
         <value><i4>1</i4></value>
        <!-- ProvisioningParameter #0.
             Hosting name is not defined -->
         <value>DomainID=</value>
         <!-- ContactDataCounter -->
         <value><i4>2</i4></value>
         <!-- Login to CP -->
         <value>LoginID=jsmith</value>
         <!-- Customer's Account ID -->
         <value>AccountID=1000002</value>
         <!-- PayToolCounter -->
         <value><i4>2</i4></value>
         <!-- Identifier of Payment tool
              registered on Customer's Account -->
         <value>PayToolID=3</value>
         <!-- IP Address -->
         <value>IPAddressID=192.168.232.10</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Account ID -->
              <value><i4>1000002</i4></value>
              <!-- Order ID -->
              <value><i4>16</i4></value>
              <!-- Login -->
              <value><string>jsmith</string></value>
              <!-- Creation Time -->
              <value><string>06-Jan-2009</string></value>
              <!-- Payment ID -->
              <value><i4>000007</i4></value>
              <!-- Order Total -->
              <value><double>85.000000</double></value>
              <!-- Order Tax Total -->
              <value><double>0.000000</double></value>
              <!-- Discount Total -->
              <value><double>0.000000</double></value>
              <!-- Order MerchTotal -->
              <value><double>85.000000</double></value>
              <!-- Order Description -->
              <value><string>
               Subscription on Plan #3 (Linux Basic) for 1 Year(s).
              </string></value>
              <!-- Order Number -->
              <value><string>SO000008</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlaceOrderForAccount_API

Method is obsolete left for backward compatibility. Refer to the PlaceOrderAndAuthorize_API method (on page 391).

PlacePlanPeriodSwitchOrder_API

	Parameters	
Description	**Passed**	**Returned**
Method places upgrade or downgrade order for plan or plan period switch. Method can be used to upgrade customer subscription from one plan to another, or to switch subscription between subscription periods of same plan. Current subscription should be active, destination subscription period should be active.	4	9

Syntax

```
ItemResult BM::PlacePlanPeriodSwitchOrder_API(
  Int SubscriptionID;
  Int PlanID;
  Int PeriodID;
  Int UpgradeStartType.
)
returns
  Int OrderID - ID of created order;
  Double Total - total of created order without taxes and discounts;
  Double TaxTotal - tax total of created order (sum of all taxes);
  Double DiscTotal - discount total of created order (sum of discount);
  Double MerchTotal - total of created order including taxes and discounts;
  Str(10) OrderNbr - order number;
  Str(4096) Descr      - order description;
  Str(40) CreationTime, FormatDateTimeSecLong - order creation date and time;
  Int DocID - ID of payment created for order. If payment is absent, method returns
"0".
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** PlanID - ID of destination plan for upgrades / downgrades from one plan to another; ID of current subscription service plan for switching between subscription periods of same plan;

- **Int** PeriodID - ID of subscription period of destination service plan or ID of another period of same service plan;
- **Int** UpgradeStartType - defines when new subscription period will be calculated from: "10" - from the start date of current subscription period, "20" - from the date of upgrade.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all  comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlacePlanPeriodSwitchOrder_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1001591</i4></value>
         <!-- PlanID -->
         <value>1</value>
         <!-- PeriodID -->
         <value>2</value>
         <!-- UpgradeStartType -->
         <value>20</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Order ID -->
              <value><i4>79</i4></value>
              <!-- Order Total -->
              <value><double>1.860000</double></value>
              <!-- Order TaxTotal -->
              <value><double>0.000000</double></value>
              <!-- Order Discount Total -->
              <value><double>0.000000</double></value>
              <!-- Order MerchTotal -->
              <value><double>1.860000</double></value>
              <!-- Order Number -->
              <value><string>DG000001</string></value>
              <!-- Order Description -->
              <value><string>
               Order on switch Subscription Period for Subscription #1000009.
              </string></value>
              <!-- Order Creation Time -->
              <value><string>20-Jun-2009 13:04:22</string></value>
              <!-- Order Payment ID -->
              <value><i4>57</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlaceResourceUpgradeOrder_API

Description	Parameters	
	Passed	**Returned**
Method places upgrade or downgrade order for resources in subscription. The method can be used only for active subscriptions and resources in the *Installed* status only. **Note:** paid in advance resource recurring fee is returned only if you refuse from additionally purchased resources. Setup fee is not a subject for a refund as well as included resource amount.	8 or more	8

Syntax

```
ItemResult BM::PlaceResourceUpgradeOrder_API(
  Int SubscriptionID;
  Int usePayTool;
  Int useDefaultPayTool;
  Int UserID;
  Str FromIP;
  Int ResourceID;
  Int NewResourceID;
  Double Amount.
)
returns
  Int NewOrderID - ID of created upgrade order;
  Double Total - total of created order without taxes and discounts;
  Double TaxTotal - tax total of created order (sum of all taxes);
  Double DiscTotal - discount total of created order (sum of discount);
  Double MerchTotal - total of created order including taxes and discounts;
  Str OrderNbr - order number;
  Str Descr- order description;
  Str CreationTime, FormatDateTimeSecLong - Unix time stamp of order creation.
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** usePayTool - ID of payment method to be used for payment. The parameter should be passed as NULL (-2147483648) if default payment method is to be used (useDefaultPayTool=1);
- **Int** useDefaultPayTool - signifies whether the default payment method to be used for payment (1- yes, 0 - no);
- **Int** UserID - ID of account the order is placed for (obsolete, though required parameter, any non-empty value can be used);

- **Str** FromIP - IP address of host the order is placed from(needed for fraud screening);

The following three parameters are always passed together, they specify the old and new resource and resource amount, thus there can be passed as many hashes of three parameters, as many resources are to be upgraded:

- **Int** ResourceID - ID of old resource;
- **Int** NewResourceID - ID of new resource;
- **Double** Amount - the amount of resource (in units) to be added or dropped.

Important: resource-to-resource upgrades and downgrades are supported in PBA versions 5.0 and higher. For earlier PBA versions upgrades and downgrades are allowed for the same resource only. Namely, ResourceID and NewResourceID values must be the same.

Example 1

The example presents request that downgrades an amount of the resource ID 1000161 on 7 units and upgrades an amount of the resource ID 1000009 on 3 units. Customer paid for one billing period in advance. Resources recurring fee is $10 per unit for the resource ID 1000161 and $3 per unit for the resource ID 1000009. Created order is to be paid with customer default payment method.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
       <name>Method</name>
        <value>PlaceResourceUpgradeOrder_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- Subscription ID -->
           <value><i4>1000149</i4></value>
           <!-- usePayTool=NULL as default payment method is to be used -->
           <value><i4>-2</i4></value>
           <!-- useDefaultPayTool=1 (yes) -->
           <value><i4>1</i4></value>
               <!-- UserID -->
           <value><i4>1001</i4></value>
           <!-- From IP -->
           <value><string>10.24.8.206</string></value>
           <!-- Resource ID -->
           <value><i4>1000161</i4></value>
           <!-- New resource ID -->
           <value><i4>1000161</i4></value>
           <!-- Amount -->
           <value><double>-7.0</double></value>
           <!-- Resource ID -->
           <value><i4>1000009</i4></value>
           <!-- New resource ID -->
           <value><i4>1000009</i4></value>
           <!-- Amount -->
           <value><double>3.0</double></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- New order ID -->
              <value><i4>28</i4></value>
              <!-- Total -->
              <value><double>-67.000000</double></value>
              <!-- Tax total -->
              <value><double>0.000000</double></value>
              <!-- Discount total -->
              <value><double>0.000000</double></value>
              <!-- Merch total -->
              <value><double>67.000000</double></value>
              <!-- Order number -->
              <value><string>DG000008</string></value>
              <!-- Order description -->
              <value><string>
               Upgrade/Downgrade of Resources for Subscription #1000001.
              </string></value>
              <!-- Order creation time -->
              <value><i4>1250260050</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

Example 2

The example presents request that upgrades an amount of the resource ID 1000161 on 2 units and an amount of the resource ID 1000009 on 1 unit. Resources setup fee is $10 per unit for the resource ID 1000161 and $3 per unit for the resource ID 1000009. Created order is to be paid with payment method ID 2.

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>PlaceResourceUpgradeOrder_API</value>
       </member>
       <member>
        <name>Params</name>
         <value>
          <array>
           <data>
            <!-- Subscription ID -->
           <value><i4>1000149</i4></value>
           <!-- usePayTool -->
           <value><i4>2</i4></value>
           <!-- useDefaultPayTool=0 (no) -->
           <value><i4>0</i4></value>
                <!-- UserID -->
           <value><i4>1001</i4></value>
           <!-- From IP -->
           <value><string>10.24.8.206</string></value>
           <!-- Resource ID -->
           <value><i4>1000161</i4></value>
           <!-- New resource ID -->
           <value><i4>1000161</i4></value>
           <!-- Amount -->
           <value><double>2.0</double></value>
           <!-- Resource ID -->
           <value><i4>1000009</i4></value>
           <!-- New resource ID -->
           <value><i4>1000009</i4></value>
           <!-- Amount -->
           <value><double>1.0</double></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- New order ID -->
              <value><i4>29</i4></value>
              <!-- Total -->
              <value><double>23.000000</double></value>
              <!-- Tax total -->
              <value><double>0.000000</double></value>
              <!-- Discount total -->
              <value><double>0.000000</double></value>
              <!-- Merch total -->
              <value><double>23.000000</double></value>
              <!-- Order number -->
              <value><string>UG000008</string></value>
              <!-- Order description -->
              <value><string>
               Upgrade/Downgrade of Resources for Subscription #1000001.
              </string></value>
              <!-- Order creation time -->
              <value><i4>1253265050</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlaceSubscriptionCancellationOrder_API

Description	Parameters	
	Passed	**Returned**
Method places cancellation order for specified subscription. **Warning!** The method allows placing cancellation order for subscriptions in any status, except for the *Ordered*, *Terminated* and *Cancelled* statuses. Placing cancellation orders for the subscriptions in mentioned statuses is prohibited in PBA GUI.	4	2

Syntax

```
ItemResult BM::PlaceSubscriptionCancellationOrder_API(
  Int SubscriptionID;
  Int CancelType;
  Int ReasonID;
  Str Descr
}
returns
  Str ErrorMessage - message after cancellation order for specified subscription
has been placed: "Cancellation Order has been placed".
  Int CancellationOrderID - The ID of placed cancellation order.
```

Special Notes

- **Int** SubscriptionID - ID of subscription to be cancelled;
- **Int** CancelType - cancellation type:
  - 10 - customer is refunded. Corresponds to the behavior of subscription, for which all checkboxes of Subscription Cancellation Options (Control Panel) are selected.
  - 20 - customer account is credited (credit memo is created);
  - 30 - customer is not refunded / credited. Corresponds to the behavior of subscription, for which all checkboxes of Subscription Cancellation Options (Control Panel) are deselected.
- **Int** ReasonID - ID of system reason code to be stated as reason of subscription cancellation. Available reason codes can be accessed from **System** > **Settings** > **Order Processing** > **Reason Codes** submenu of the Navigation tree;
- **Str** Descr - optional comment on the reason subscription is cancelled.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlaceSubscriptionCancellationOrder_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>123</i4></value>
         <!-- CancelType  -->
         <value><i4>10</i4></value>
         <!-- ReasonID -->
         <value><i4>1</i4></value>
         <!-- Descr -->
         <value>Subscription cancellation due to customer request</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value><string>Cancellation order has been placed.
</string></value>
                    <value><i4>10014</i4></value>
             </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlanCategoryDetailsGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns properties of specified plan category.	1	4

Syntax

```
ItemResult BM::PlanCategoryDetailsGet_API(
  Int planCategoryID.
)
returns
  Int planCategoryID - plan category ID;
  Str(30) name - plan category name;
  Str(1024) description - plan category description;
  Int AccountID - account ID of plan category owner.
```

Special Notes

Int planCategoryID - ID of plan category.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanCategoryDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- planCategoryID -->
         <value><i4>2</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
          <array>
           <data>
            <value>
            <array>
             <data>
              <!-- Plan Category ID -->
              <value><i4>2</i4></value>
              <!-- Plan Category Name -->
              <value><string>Linux Hosting</string></value>
              <!-- Plan Category Description -->
              <value><string>Linux Hosting category</string></value>
              <!-- Account ID -->
              <value><i4>1</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanCategoryListGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of plan categories for specified vendor account.	2	3

Syntax

```
ListResult BM::PlanCategoryListGet_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int PlanCategoryID - plan category ID;
  Str(30) Name - plan category name;
  Str(1024) Description - plan category description.
```

Special notes

- **Int** AccountID - vendor account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (PlanCategoryID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanCategoryListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Plan Category Name -->
                 <value><string>Domains</string></value>
                 <!-- Plan Category Description -->
                 <value><string>Domain service plans</string></value>
                </data>
               </array>
              </value>
               <value>
               <array>
                <data>
                 <!-- Plan Category ID -->
                 <value><i4>2</i4></value>
                 <!-- Plan Category Name -->
                 <value><string>Linux Hosting</string></value>
                 <!-- Plan Category Description -->
                 <value><string>Linux Hosting category</string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanDetailsGet_API

Description	Parameters	
	Passed	Returned
Method returns details for specified service plan.	1	16

Syntax

```
ItemResult BM::PlanDetailsGet_API(
  Int PlanID.
)
returns
  Int PlanID - requested service plan ID;
  Str(60) Name - service plan name;
  Int CategoryID - ID of the plan category assigned to the requested service plan;
  Str ResourceCurrencyID - ID of the default provider's currency;
  Str(1024) ShortDescription - requested service plan short description;
  Str(1024) LongDescription - requested service plan long description.
  Str(40) GateName - type of service. Identified by name of gate service is
provisioned through (DOMAINGATE, PEMGATE, etc);
  Int GroupID - ID of the unique group assigned to the requested service plan.
  Int IsParentReq - signifies whether parent subscription is required for this
type of service (0 - No, 1 - Yes). Some domain subscriptions require hosting
subscription to be specified as parent one;
  Int RecurringType - signifies how recurring fee is charged for subscription (10
- before billing period; 20 - after billing period; 30 - before subscription
period; 40 - at the end of month);
  Int BillingPeriodType - subscription billing period type (4 - monthly on
statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
  Int BillingPeriod - subscription billing period duration (if BillingPeriodType=2
and BillingPeriod=1, subscription billing period is 1 month);
  Int ShowPriority - signifies the priority in which the service plan is shown
among other ones of the same sales category in the online store. The highest
priority is 1, thus, service plan with such a priority is shown in the topmost
left position in the online store;
  Int Default_PlanPeriodID - ID of the subscription period that is automatically
selected when customer orders the plan in online store.
  Int IsOTFI - signifies whether services of this plan are sold without
subscription creation (0 - No, 1 - Yes), namely whether the plan is one time fee
one.
  Str DocID - ID of the image blob document in a special storage (reference to an
image loaded for a service plan).
```

Special Notes

Int PlanID - ID of the requested service plan.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>3</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- PlanID -->
              <value><i4>3</i4></value>
              <!-- Plan Name -->
              <value><string>Linux Basic</string></value>
              <!-- Plan Category ID -->
              <value><i4>2</i4></value>
              <!-- Currency ID -->
              <value><string>USD</string></value>
              <!-- Short Description -->
              <value><string>Get Hosting!!!</string></value>
              <!-- Long Description -->
              <value><string>Get Hosting!!!</string></value>

              <!-- Gate Name -->
              <value><string>DUMMYGATE</string></value>

              <!-- Unique Group ID -->
              <value><i4>-2</i4></value>

              <!-- IsParentRequired -->
              <value><i4>0</i4></value>

              <!-- RecurringType -->
              <value><i4>10</i4></value>

              <!-- BillingPeriodType -->
              <value><i4>2</i4></value>

              <!-- BillingPeriod -->
              <value><i4>1</i4></value>

              <!-- Show Priority -->
              <value><i4>2</i4></value>

              <!-- Default PlanPeriod ID -->
              <value><i4>3</i4></value>

              <!-- IsOne Time Fee Item -->
              <value><i4>0</i4></value>

              <!-- Blob Document ID -->
              <value><string>65</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListGet_API

Description	Parameters	
	Passed	Returned
Method returns details of specified service plans.	1 or more	15

Syntax

```
ListResult BM::PlanListGet_API(
  Int PlanID_1;
  ...
  Int PlanID_N.
)
returns
  Int PlanID - service plan ID;
  Str Name - service plan name;
  Str ShortDescription - service plan short description;
  Str LongDescription - service plan long description;
  Int PlanCategoryID - ID of service plan plan category;
  Str STType - name of service gate the service plan is provisioned through;
  Int IsPublished - signifies if service plan is published or not, disabled by its
service template or by provider:
  "0" - published,
  "1" - not published,
  "2" - disabled by Template,
  "3" - not published and disabled by template;
  "4" - disabled by provider;
  "5" - not published and disabled by provider;
  "6" - disabled by provider and by template;
  "7" - not published and disabled by provider and by template;
  Int GroupID - ID of unique group service plan belongs to;
  Int RecurringType - service plan recurring fee collection method (Charge for
Subscription parameter): "10" - before billing period; "20" - after billing period;
"30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - service plan billing period type: "2" - billing period is
a fixed number of months; "3" - billing period is a fixed number of years; "4" -
subscription is billed monthly on statement cycle date;
  Int BillingPeriod - service plan billing period;
  Int IsParentReq - signifies if parent subscription is required for this type of
service (0 - No, 1 - Yes). Parent subscription is required if service plan is
sold as up-sale to another one;
  Int ShowPriority - order number of service plan in online store;
  Int Default_PlanPeriodID - ID of default subscription period of service plan.
Used in online store.
  Int IsOTFI - signifies whether services of this plan are sold without
subscription creation (0 - No, 1 - Yes), namely whether the plan is one time fee
one.
```

Special Notes

Int PlanID - requested service plan ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
<methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>PlanListGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
         <!-- PlanID  1 -->
          <value><i4>1</i4></value>
         <!-- PlanID  2 -->
          <value><i4>3</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- PlanID -->
                 <value><i4>1</i4></value>
                 <!-- Plan Name -->
                 <value><string>.biz</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .biz zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .biz zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Service Template Type -->
                 <value><string>DOMAINGATE</string></value>
                 <!-- IsPublished -->
                 <value><i4>0</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- IsParentRequired -->
                 <value><i4>0</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                 <!-- Default Plan Period -->
                 <value><i4>2</i4></value>
                 <!-- IsOTFI -->
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
   ...
              <value>
               <array>
```

```
                <data>
                 <!-- PlanID -->
                 <value><i4>3</i4></value>
                 <!-- Plan Name -->
                 <value><string>Linux Basic</string></value>
                 <!-- Plan Category ID -->
                 <value><i4>2</i4></value>
                 <!-- Currency ID -->
                 <value><i4>1</i4></value>
                 <!-- Short Description -->
                 <value><string>Get Hosting!!!</string></value>
                 <!-- Long Description -->
                 <value><string>Get Hosting!!!</string></value>
                 <!-- Gate Name -->
                 <value><string>DUMMYGATE</string></value>
                 <!-- Unique Group ID -->
                 <value><i4>-2</i4></value>
                 <!-- IsParentRequired -->
                 <value><i4>0</i4></value>
                 <!-- RecurringType -->
                 <value><i4>10</i4></value>
                 <!-- BillingPeriodType -->
                 <value><i4>2</i4></value>
                 <!-- BillingPeriod -->
                 <value><i4>1</i4></value>
                 <!-- Show Priority -->
                 <value><i4>2</i4></value>
                 <!-- Default PlanPeriod ID -->
                 <value><i4>3</i4></value>
                 <!-- IsOTFI -->
                <value><i4>1</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlanListAvailableGet_API

Description	Parameters	
	Passed	**Returned**

Method returns the list of published service plans of specified plan category. Result can be truncated depending on whether "STType" parameter is passed: • parameter passed - method returns all published service plans of specified plan category provisioned through specified service gate; • not passed - method returns all published service plans within specified plan category.	4	14

Syntax

```
ListResult BM::PlanListAvailableGet_API(
  Int AccountID;
  Str(40) STType;
  Int PlanCategoryID;
  Int SortNo.
)
returns
  Int PlanID - service plan ID;
  Str Name - service plan name;
  Str ShortDescription - service plan short description;
  Str LongDescription - service plan long description;
  Int PlanCategoryID - ID of service plan plan category;
  Str STType - name of service gate the service plan is provisioned through;
  Int IsPublished - signifies if service plan is published or not, disabled by its
service template or by provider:
  "0" - published,
  "1" - not published,
  "2" - disabled by Template,
  "3" - not published and disabled by template;
  "4" - disabled by provider;
  "5" - not published and disabled by provider;
  "6" - disabled by provider and by template;
  "7" - not published and disabled by provider and by template;
  Int GroupID - ID of unique group service plan belongs to;
  Int RecurringType - service plan recurring fee collection method (Charge for
Subscription parameter): "10" - before billing period; "20" - after billing period;
"30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - service plan billing period type: "2" - billing period is
a fixed number of months; "3" - billing period is a fixed number of years; "4" -
subscription is billed monthly on statement cycle date;
  Int BillingPeriod - service plan billing period;
  Int IsParentReq - signifies if parent subscription is required for this type of
service. Parent subscription is required if service plan is sold as up-sale to
another one;
  Int ShowPriority - order number of service plan in online store;
  Int Default_PlanPeriodID - ID of default subscription period of service plan.
Used in online store.
```

Special Notes

- **Int** AccountID - account ID of plan category owner;

- **Str** STType - name of service gate service plans are provisioned through: PEMGATE, DOMAINGATE, DUMMYGATE, etc. This parameter can be determined as follows: 1) Open properties of one of service plans included in desired plan category 2) Enter service template properties 3) Enter service template service gate properties 4) **Container Name** parameter here is to be passed as *STType*. If parameter is passed empty, method returns all published service plans of specified category.
- **Int** planCategoryID - plan category ID;
- **Int** SortNo - parameter defines how output data is sorted: "1" - the output is sorted by plan ID in ascending order, "2" - by plan name in ascending order, etc. Negative value makes output sorted in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
<methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>PlanListAvailableGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- AccountID -->
           <value><i4>1</i4></value>
           <!-- STType -->
           <value>DOMAINGATE</value>
           <!-- planCategoryID -->
           <value><i4>1</i4></value>
           <!-- SorNo -->
           <value><i4>1</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- PlanID -->
                 <value><i4>1</i4></value>
                 <!-- Plan Name -->
                 <value><string>.biz</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .biz zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .biz zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Service Template Type -->
                 <value><string>DOMAINGATE</string></value>
                 <!-- IsPublished -->
                 <value><i4>0</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- IsParentRequired -->
                 <value><i4>0</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                 <!-- Default Plan Period -->
                 <value><i4>2</i4></value>
                </data>
               </array>
              </value>
   ...
             <value>
              <array>
                <data>
                 <!-- PlanID -->
                 <value><i4>2</i4></value>
                 <!-- Plan Name -->
                 <value><string>.com</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .com zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .com zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Service Template Type -->
                 <value><string>DOMAINGATE</string></value>
                 <!-- IsPublished -->
                 <value><i4>0</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- IsParentRequired -->
                 <value><i4>0</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                 <!-- Default Plan Period -->
                 <value><i4>2</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListAvailableUpsaleGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of available up-sale plans for specified service plan. Service gate and plan category of up-sale service plan can be specified additionally.	4	10

Syntax

```
ListResult BM::PlanListAvailableUpsaleGet_API(
  Int PlanID;
  Str(40) STType;
  Int PlanCategoryID;
  Int SortNo.
)
returns
  Int PlanID - up-sale service plan ID;
  Str(40) NameDependencePlanID - up-sale service plan name;
  Str(40) ShortDependencePlanID - up-sale service plan short description;
  Str(40) LongDependencePlanID - up-sale service plan full description;
  Int PlanCategoryID - plan category ID, up-sale service plan included in;
  Int GroupID - unique group ID, up-sale service plan included in;
  Int RecurringType - up-sale service plan recurring fee collection method (Charge
for Subscription parameter): "10" - before billing period; "20" - after billing
period; "30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - up-sale service plan billing period type: "2" - billing
period is a fixed number of months; "3" - billing period is a fixed number of
years; "4" - subscription is billed monthly on statement cycle date;
  Int BillingPeriod - subscription billing period duration (if BillingPeriodType=2
and BillingPeriod=1, subscription billing period is 1 month);
  Int ShowPriority - defines the priority in which the up-sale service plan is
shown among other ones of the same sales category in the online store. The
highest priority is 1, thus, service plan with such a priority is shown in the
topmost left position in the online store.
```

Special Notes

- **Int** PlanID - parent service plan ID;
- **Str(40)** STType - name of service gate the up-sale service plans are provisioned through. To get up-sales provisioned through any service gate, pass an empty string;
- **Int** PlanCategoryID - plan category ID, up-sale service plan included in. To get up-sales with any plan categories assigned, pass 0;
- **Int** SortNo.- defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.Type of service.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanListAvailableUpsaleGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>3</i4></value>
         <!-- STType -->
         <value>DOMAINGATE</value>
         <!-- PlanCategoryID -->
         <value><i4></i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
```

```
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Up-sale PlanID -->
                 <value><i4>1</i4></value>
                 <!-- Up-sale Plan Name -->
                 <value><string>.biz</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .biz zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .biz zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                </data>
               </array>
              </value>
...
             <value>
              <array>
                <data>
                 <!--  Up-sale PlanID -->
                 <value><i4>2</i4></value>
                 <!-- Up-sale Plan Name -->
                 <value><string>.com</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .com zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .com zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListUpsaleGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of available up-sale plans for specified service plan.	5	12

Syntax

```
ListResult BM::PlanListUpsaleGet_API(
  Int PlanID;
  Int SortNo.
)
returns
  Int PlanID - ID of up-sale service plan;
  Str(60) Name - up-sale service plan name;
  Str(1024) ShortDescription - up-sale service plan short description;
  Str(1024) LongDescription - up-sale service plan full description;
  Int CategoryID - sales category ID up-sale service plan belongs to;
  Int GroupID - ID of unique group service plan is included in;
  Int ShowPriority - defines the priority in which the service plan is shown among
other ones of the same sales category in the online store. The highest priority
is 1, thus, service plan with such a priority is shown in the topmost left
position in the online store;
  Int ChildLimit - maximum number of up-sale subscriptions set for up-sale
category (Maximum Up-sale Subscriptions parameter).
```

Special Notes

- **Int** PlanID - up-sale service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanListUpsaleGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>3</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Up-sale plan ID -->
                 <value><i4>4</i4></value>
                 <!-- Up-sale plan name -->
                 <value><string>.com</string></value>
                 <!-- Up-sale plan short description -->
                 <value><string>
                  Get a domain in .com zone!
                 </string></value>
                 <!-- Up-sale plan long description -->
                 <value><string>
                  Get a domain in .com zone for $5/month!
                 </string></value>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>1</i4></value>
                 <!-- ShowPriority -->
                 <value><i4>3</i4></value>
                 <!-- Child limit -->
                 <value><i4>6</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListUpsaleGetForSubscription_API

Description	Parameters	
	Passed	Returned
Method returns the list of available up-sale plans for specified subscription.	5	12

Syntax

```
ListResult BM::PlanListUpsaleGetForSubscription_API(
  Int SubscriptionID_1;
  ...
  Int SubscriptionID_N.
)
returns
  Int SubscriptionID - ID of requested subscription;
  Int PlanID - ID of up-sale service plan;
  Str(60) Name - up-sale service plan name;
  Str(1024) ShortDescription - up-sale service plan short description;
  Str(1024) LongDescription - up-sale service plan full description;
  Int CategoryID - sales category ID up-sale service plan belongs to;
  Int GroupID - ID of unique group service plan is included in;
  Int ShowPriority - defines the priority in which the service plan is shown among
other ones of the same sales category in the online store. The highest priority
is 1, thus, service plan with such a priority is shown in the topmost left
position in the online store;
  Int UpsellsLeft - number of available subscriptions left. It is calculated as
maximum number of up-sale subscriptions set for up-sale category (Maximum Up-sale
Subscriptions parameter) minus up-sale subscriptions sold.
```

Special Notes

Int SubscriptionID - subscription ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
```

```
      <name>Method</name>
      <value>PlanListUpsaleGetForSubscription_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID  1 -->
         <value><i4>1000009</i4></value>
         <!-- SubscriptionID  2 -->
         <value><i4>1000034</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- SubscriptionID  1 -->
                 <value><i4>1000009</i4></value>
                 <!-- Up-sale service plan ID -->
                 <value><i4>4</i4></value>
                 <!-- Up-sale service plan name -->
                 <value><string>.com</string></value>
                 <!-- Up-sale plan short description -->
                 <value><string>
                  Get a domain in .com zone!
                 </string></value>
                 <!-- Up-sale plan long description -->
                 <value><string>
                  Get a domain in .com zone for $5/month!
                 </string></value>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>1</i4></value>
                 <!-- ShowPriority -->
                 <value><i4>3</i4></value>
                 <!-- UpsellsLeft -->
                 <value><i4>6</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListAvailableUpsales4SalesCategoryGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of available up-sale plans for specified plan and sales category.	5	12

Syntax

```
ListResult BM::PlanListAvailableUpsales4SalesCategoryGet_API(
  Int PlanID;
  Str(40) STType;
  Int CategoryID;
  Int AccountID;
  Int SortNo.
)
returns
  Int PlanID - up-sale service plan ID;
  Str(60) - up-sale service plan name;
  Str(1024) ShortDescription - up-sale service plan short description;
  Str(1024) LongDescription - up-sale service plan full description;
  Int CategoryID - sales category ID;
  Int GroupID - unique group ID, up-sale service plan included in;
  Int Included - always returns "0". The parameter is obsolete, left for
compatibility with PBA versions 4.2.x and earlier;
  Int RecurringType - up-sale service plan recurring fee collection method (Charge
for Subscription parameter): "10" - before billing period; "20" - after billing
period; "30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - up-sale service plan billing period type: "2" - billing
period is a fixed number of months; "3" - billing period is a fixed number of
years; "4" - subscription is billed monthly on statement cycle date;
  Int BillingPeriod - subscription billing period duration (if BillingPeriodType=2
and BillingPeriod=1, subscription billing period is 1 month);
  Int ShowPriority - signifies the priority in which the up-sale service plan is
shown among other ones of the same sales category in the online store. The
highest priority is 1, thus, service plan with such a priority is shown in the
topmost left position in the online store;
  Int ChildLimit - maximum number of up-sale subscriptions set for up-sale
category (Maximum Up-sale Subscriptions parameter).
```

Special Notes

- **Int** PlanID - parent service plan ID;
- **Str(40)** STType - name of service gate the up-sale service plans are provisioned through. To get up-sales provisioned through any service gate, pass an empty string;
- **Int** CategoryID - sales category ID, assigned to parent plan as an up-sale category. To get up-sales of all sales categories assigned, pass 0;
- **Int** AccountID - vendor account ID;

- **Int** SortNo.- defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanListAvailableUpsales4SalesCategoryGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>3</i4></value>
         <!-- STType -->
         <value>DOMAINGATE</value>
         <!-- CategoryID -->
         <value><i4>1</i4></value>
         <!-- AccountID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
```

```
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Plan ID -->
                 <value><i4>4</i4></value>
                 <!-- Plan name -->
                 <value><string>.com</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .com zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .com zone for $5!
                 </string></value>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Included -->
                 <value><i4>0</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- ShowPriority -->
                 <value><i4>3</i4></value>
                 <!-- Child limit -->
                 <value><i4>6</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanListAvailable4SalesCategoryGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of published plans included into specified sales category.	4	16

Syntax

```
ListResult BM::PlanListAvailable4SalesCategoryGet_API(
  Int AccountID;
  Str(40) STType;
  Int CategoryID;
  Int SortNo.
)
returns
  Int PlanID - ID of service plan included in the requested sales category;
  Str(60) Name - service plan name;
  Str(1024) ShortDescription - service plan short description;
  Str(1024) LongDescription - service plan full description;
  Int CategoryID - requested sales category ID;
  Str STType - service plan provisioning gate, for example, DOMAINGATE, PEMGATE,
DUMMYGATE etc.;
  Int IsPublished - signifies if service plan is published: "0" - published, "1" -
not published. Method returns only published plan, so the parameter is always
"0";
  Int GroupID - ID of unique group service plan is included in;
  Int RecurringType - service plan recurring fee collection method (Charge for
Subscription parameter): "10" - before billing period; "20" - after billing period;
"30" - before subscription period; "40" - in the end of month;
  Int BillingPeriodType - service plan billing period type: "2" - billing period
is a fixed number of months; "3" - billing period is a fixed number of years; "4"
- subscription is billed monthly on statement cycle date;
  Int BillingPeriod - subscription billing period duration (if BillingPeriodType=2
and BillingPeriod=1, subscription billing period is 1 month);
  Int IsParentReq - signifies whether parent subscription is required for this
type of service (0 - No, 1 - Yes). Some domain subscriptions require hosting
subscription to be specified as parent one;
  Int ShowPriority - signifies the priority in which the service plan is shown
among other ones of the same sales category in the online store. The highest
priority is 1, thus, service plan with such a priority is shown in the topmost
left position in the online store;
  Int Default_PlanPeriod - ID of the subscription period that is automatically
selected when customer orders the plan in online store.
  Int IsOTFI - signifies whether services of this plan are sold without
subscription creation (0 - No, 1 - Yes), namely whether the plan is one time fee
one.
  Str DocID - ID of the image blob document in a special storage (reference to an
image loaded for a service plan).
```

Special Notes

- **Int** AccountID - vendor account ID (provider or reseller);

- **Str(40)** STType - service plans provisioning gate, for example, DOMAINGATE, PEMGATE, DUMMYGATE etc. The parameter is optional. If STType is passed, the method returns service plans provisioned by specified service gate only;
- **Int** CategoryID - sales category ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanListAvailable4SalesCategoryGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1</i4></value>
         <!-- STType -->
         <value>DOMAINGATE</value>
         <!-- CategoryID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
```

```
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- PlanID -->
                 <value><i4>1</i4></value>
                 <!-- Plan Name -->
                 <value><string>.biz</string></value>
                 <!-- Short Description -->
                 <value><string>
                  Get a domain in the .biz zone!
                 </string></value>
                 <!-- Long Description -->
                 <value><string>
                  Get a domain in the .biz zone for $5!
                 </string></value>
                 <!-- Plan Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Service Template Type -->
                 <value><string>DOMAINGATE</string></value>
                 <!-- IsPublished -->
                 <value><i4>0</i4></value>
                 <!-- Unique Group ID -->
                 <value><i4>5</i4></value>
                 <!-- Recurring Type -->
                 <value><i4>10</i4></value>
                 <!-- Billing Period Type -->
                 <value><i4>4</i4></value>
                 <!-- Billing Period -->
                 <value><i4>1</i4></value>
                 <!-- IsParentRequired -->
                 <value><i4>0</i4></value>
                 <!-- Show Priority -->
                 <value><i4>4</i4></value>
                 <!-- Default Plan Period -->
                 <value><i4>2</i4></value>
                 <!-- IsOne Time Fee Item -->
                 <value><i4>0</i4></value>
                     <!-- Blob Document ID -->
                      <value><string>7f0be396a6af295f</string></value>
                </data>
               </array>
              </value>
   ...
              <value>
               <array>
                <data>
                 <!-- PlanID -->
                 <value><i4>3</i4></value>
                 <!-- Plan Name -->
                 <value><string>Linux Basic</string></value>
                 <!-- Plan Category ID -->
                 <value><i4>2</i4></value>
                 <!-- Currency ID -->
                 <value><i4>1</i4></value>
                 <!-- Short Description -->
                 <value><string>Get Hosting!!!</string></value>
```

```
                  <!-- Long Description -->
                  <value><string>Get Hosting!!!</string></value>

                  <!-- Gate Name -->
                  <value><string>DUMMYGATE</string></value>

                  <!-- Unique Group ID -->
                  <value><i4>-2</i4></value>

                  <!-- IsParentRequired -->
                  <value><i4>0</i4></value>

                  <!-- RecurringType -->
                  <value><i4>10</i4></value>

                  <!-- BillingPeriodType -->
                  <value><i4>2</i4></value>

                  <!-- BillingPeriod -->
                  <value><i4>1</i4></value>

                  <!-- Show Priority -->
                  <value><i4>2</i4></value>

                  <!-- Default PlanPeriod ID -->
                  <value><i4>3</i4></value>

                  <!-- IsOne Time Fee Item -->
                  <value><i4>1</i4></value>

                      <!-- Blob Document ID -->
                       <value><string>7f0be396a6af295f</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlanParamsGetAndValidate_API

Description	Parameters	
	Passed	**Returned**

Method pulls a list of service template (ST) service parameters for validation and returns full list of ST parameters with validation result. The list of ST parameters differs depending on template type: POA hosting template, domain template, licensed services template. To get the full list of service parameters, invoke method with two parameters passed: service plan ID and ST ID. **Note**: if some exception occurs when processing submitted values, method returns base64 encoded message instead of standard data packet. This method can be used for validation of subscription provisioning parameters submitted through a third-party online store or for migration purposes.	2 or more	9 or more

Syntax

```
ListResult BM::PlanParamsGetAndValidate_API(
  Int PlanID;
  Int ServiceTemplateID;
  Str(40) STParameterID_1=ParameterValue;
  ...
  Str(40) STParameterID_N=ParameterValue.
)
returns
  Str(40) ParamID - ID of ST service parameter;
  Str(40) ParamType - type ST of service parameter;
  Str(40) ParamName - name of ST service parameter.
  Str(40) Description - description of ST service parameter;
  Str(40) DefaultValue - if parameter has default value, returns this value. For
example, default value for 'OrderOperationType' parameter of domain ST is '10'
(registration);
  Int Required - signifies if parameter is required: '1' - required, '0' - not
required;
  Int Status - parameter validation status:
'0' - parameter is valid;
'1' - parameter value is absent and need to be asked;
'2' - parameter is invalid (see details in Message slot);
'3' - parameter value is not needed for service provisioning;
  Str(256) Message - parameter validation result;
  Int ST_ParamWeight - service parameter sort order in online store and CP.
```

Special Notes

- **Int** PlanID - ID of service plan1;
- **Int** ServiceTemplateID - ID of service template the plan is based on;
- **Str(40)** STParameterID_1=ParameterValue - ST required parameters are submitted in the following form: "ParameterID=ParameterValue", for example "DomainID=domainname".

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanParamsGetAndValidate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>16</i4></value>
         <!-- ServiceTemplateID -->
         <value>1000012</value>
         <!-- STParameterID  1=ParameterValue -->
         <value>DomainID=iwantdomain</value>
         <!-- STParameterID  2=ParameterValue -->
         <value>OrderOperationType=10</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                <!-- Parameter ID -->
                <value><string>DomainID</string></value>
                <!-- Parameter type -->
                <value><string>DOMAIN_NAME</string></value>
                <!-- Parameter name -->
                <value><string>Domain name</string></value>
                <!-- Parameter description -->
                <value><string>Domain name</string></value>
                <!-- Default Value -->
                <value><string>iwantdomain</string></value>
                <!-- Required -->
                <value><string>1</string></value>
                <!-- Parameter validation status -->
                <value><string>0</string></value>
                <!-- Message -->
                <value><string>Everything is ok</string></value>
                <!-- Parameter weight -->
                <value><i4>1</i4></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
                <!-- Parameter ID -->
                <value><string>OrderOperationType</string></value>
                <!-- Parameter type -->
                <value><string>ORDER_OPERATION_TYPE</string></value>
                <!-- Parameter name -->
                <value><string>Domain Operation</string></value>
                <!-- Parameter description -->
                <value><string>
                 What to do with Domain.
                </string></value>
                <!-- Default Value -->
                <value><string>10</string></value>
                <!-- Required -->
                <value><string>1</string></value>
                <!-- Parameter validation status -->
                <value><string>0</string></value>
                <!-- Message -->
                <value><string></string></value>

                <!-- Parameter weight -->
                <value><i4>2</i4></value>
              </data>
             </array>
            </value>
            <value>
             <array>
              <data>

                <!-- Parameter ID -->
                <value><string>TransferKey</string></value>

                <!-- Parameter type -->
                <value><string>STR</string></value>

                <!-- Parameter name -->
                <value><string>Transfer key</string></value>

                <!-- Parameter description -->
                <value><string>
                 Password of Domain you transfering.
                </string></value>

                <!-- Default Value -->
                <value><string></string></value>

                <!-- Required -->
                <value><string>0</string></value>

                <!-- Parameter validation status -->
                <value><string>1</string></value>

                <!-- Message -->
                <value><string>Just</string></value>

                <!-- Parameter weight -->
                <value><i4>3</i4></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlanParentPlansAvailableListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of plans that selected plan could be up-sold to.	2	6

Syntax

```
ListResult BM::PlanParentPlansAvailableListGet_API(
  Int PlanID;
  Int SortNo.
)
returns
  Int PlanID - parent service plan ID;
  Str(60) Name - parent service plan name;
  Str(1024) ShortDescription - parent service plan short description;
  Str(1024) LongDescription - parent service plan long description;
  Int PlanCategoryID - ID of sales category up-sale service plan assigned to;
  Int ChildLimit - maximum number of up-sale subscriptions set for up-sale
category (Maximum Up-sale Subscriptions parameter).
```

Special Notes

- **Int** PlanID - up-sale service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanParentPlansAvailableListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>4</i4></value>
         <!-- SortNo -->
         <value><i4>2</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                <!-- Parent PlanID -->
                <value><i4>3</i4></value>
                <!-- Parent Plan name -->
                <value><string>Linux Basic</string></value>
                <!-- Parent Plan short description -->
                <value><string>Get Hosting!!!</string></value>
                <!-- Parent Plan long description -->
                <value><string>Get Hosting!!!</string></value>
                <!-- Plan category ID -->
                <value><i4>1</i4></value>
                <!-- Child limit -->
               <value><i4>6</i4></value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
 </methodResponse>
```

PlanPeriodListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of subscription periods configured under specified service plan.	2	17

Syntax

```
ListResult BM::PlanPeriodListGet_API(
        Int PlanID - service plan ID.
        Int SortNo - defines output order.
)
returns
        Int PlanPeriodID - subscription period ID.
        Int Period - subscription period duration.
        Int PeriodType - subscription period type: "1" - days, "2" - months, "3" -
years.
        Int Trial - signifies if subscription period is trial ("1") or not ("0").
        Double SetupFee - subscription period setup fee.
        Double SubscriptionFee - subscription period recurring fee.
        Double RenewalFee - subscription period renewal fee.
        Double TransferFee - subscription period transfer fee (for domain service
plans).
        Double NonRefundableAmount - amount not refunded to customer when refund is
claimed (on subscription cancellation, for example).
        Int RefundPeriod - during this period (in days) customer may claim a full
money back (minus non refundable amount).
        Int Enabled - signifies if period is active ("1") or not ("0").
        Double NumberOfPeriods - number of billing periods within subscription
period.
        Str(1024) FeeText - the text displayed as description of period fees, for
example "$9.99 setup + $18/month".
        Int SortNumber - order in which subscription periods are shown in online
store and CP: "1" - on top, "2" - below it, etc.
        Int IsOTFI - signifies whether services of this plan are sold without
subscription creation (0 - No, 1 - Yes), namely whether the plan is one time fee
one.
        Double DepositFee - the deposit amount that is included in the Sales Order
total amount and is displayed as a separate detail. When the Order is placed the
deposit is credited to the customer account and is displayed on the customer
balance in the Control Panel. This deposit is later used as a guaranteed payment.
In case of a deposit overdraft, the customer is asked to make an additional
payment.
        Str(1024) DepositDescr - the text displayed as description of the deposit
fee.
```

Special Notes

- **Int** `SortNo`. 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (PlanPeriod ID) in descending order.
- **Double** `NumberOfPeriods`. This parameter is calculated automatically and depends on **Billing Period Type** parameter of service plan. For example, if **Billing Period Type** is "Fixed Number of Months" and subscription period is 2 years long, there will be 24 billing periods, meaning customer will be charged recurring fee 24 times.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanPeriodListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>4</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Plan period ID -->
                 <value><i4>5</i4></value>
                 <!-- Period duration -->
                 <value><i4>1</i4></value>
                 <!-- Period type -->
                 <value><i4>3</i4></value>
                 <!-- Trial -->
                 <value><i4>0</i4></value>
                 <!-- Setup Fee -->
                 <value><double>1.000000</double></value>
                 <!-- Subscription Fee -->
                 <value><double>2.000000</double></value>
                 <!-- Renewal Fee -->
                 <value><double>0.000000</double></value>
                 <!-- Transfer Fee -->
                 <value><double>0.000000</double></value>
                 <!-- NonRefundable Amount -->
                 <value><double>0.000000</double></value>
                 <!-- Refund period -->
                 <value><i4>0</i4></value>
                 <!-- Enabled -->
                 <value><i4>1</i4></value>
                 <!-- Number of periods -->
                      <value><double>3.000000</double></value>
                 <!-- Fee text -->
                 <value><string></string></value>
                 <!-- Sort number -->
                 <value><i4>1</i4></value>
                 <!-- Is One Time Fee Item -->
                 <value><i4>0</i4></value>
                 <!-- DepositFee -->
                 <value><double>5</double></value>
                 <!-- DepositDescr -->
                 <value><string>Reseller Domain Deposit</string></value>
               </data>
              </array>
             </value>
             <value>
              <array>
               <data>
```

```
                <!-- Plan period ID -->
                <value><i4>6</i4></value>

                <!-- Period duration -->
                <value><i4>2</i4></value>

                <!-- Period type -->
                <value><i4>3</i4></value>

                <!-- Trial -->
                <value><i4>0</i4></value>

                <!-- Setup Fee -->
                <value><double>2.000000</double></value>

                <!-- Subscription Fee -->
                <value><double>5.000000</double></value>

                <!-- Renewal Fee -->
                <value><double>0.000000</double></value>

                <!-- Transfer Fee -->
                <value><double>0.000000</double></value>

                <!-- NonRefundable Amount -->
                <value><double>0.000000</double></value>

                <!-- Refund period -->
                <value><i4>0</i4></value>

                <!-- Enabled -->
                <value><i4>1</i4></value>

                <<!-- Number of periods -->
                     <value><double>24.000000</double></value>

                <!-- Fee text -->
                <value><string></string></value>

                <!-- Sort number -->
                <value><i4>2</i4></value>

                <!-- Is One Time Fee Item -->
                <value><i4>0</i4></value>

                 <!-- DepositFee -->
                 <value><double>5</double></value>

                 <!-- DepositDescr -->
                 <value><string>Reseller Domain Deposit</string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

PlanPeriodDetailsGet_API

Description	Parameters	
	Passed	Returned

Method returns details for specified subscription period.	1	17

Syntax

```
ItemResult BM::PlanPeriodDetailsGet_API(
  Int PlanPeriodID.
)
returns
  Int PlanPeriodID - service plan subscription period ID;
  Int PlanID - service plan ID;
  Int Period - subscription period duration (for example, 1 if subscription period
is 1 year);
  Int PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 -
Month(s), 3 - Year(s);
  Int Trial - whether subscription period is trial (free): 0 - No, 1 - Yes;
  Double SetupFee - subscription setup fee. It is initial and one time charge
which is taken for subscription for this plan;
  Double SubscriptionFee - subscription recurring fee. It is taken once a billing
period;
  Double RenewalFee - subscription renewal fee. It is charged for subscription
renewal;
  Double TransferFee - subscription transfer fee. It is charged for domain
transfer and relevant for domain plans only;
  Double NonRefundableAmt - subscription non-refundable amount. The sum is not
refunded to customer when refund is claimed (on subscription cancellation, for
example);
  Int RefundPeriod - subscription refund period duration in days (for example, 5
if refund period is 5 days). During refund period customer can cancel his
subscription and get all money back, except non-refundable amount;
  Int Enabled - whether subscription period is active : 0 - No, 1 - Yes;
  Int NumberOfPeriods - number of billing periods within subscription period. This
parameter is calculated automatically and depends on Billing Period Type parameter of
service plan. For example, if Billing Period Type is "Fixed Number of Months" and
subscription period is 2 years long, there will be 24 billing periods, meaning
customer will be charged recurring fee 24 times;
  Str(1024) FeeText - text displayed as description of period fees, for example
"$9.99 setup + $18/month";
  Int SortNumber - order in which subscription periods are shown in online store
and CP: "1" - on top, "2" - below it, etc.
  Double DepositFee - the deposit amount that is included in the Sales Order total
amount and is displayed as a separate detail. When the Order is placed the
deposit is credited to the customer account and is displayed on the customer
balance in the Control Panel. This deposit is later used as a guaranteed payment.
In case of a deposit overdraft, the customer is asked to make an additional
payment.
  Str DepositFeeText - text displayed as description of deposit fees.
```

Special Notes

Int PlanPeriodID - ID of requested subscription period.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanPeriodDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanPeriodID -->
         <value><i4>5</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Plan period ID -->
                 <value><i4>5</i4></value>
                 <!-- PlanID -->
                 <value><i4>12</i4></value>
                 <!-- Period duration -->
                 <value><i4>1</i4></value>
                 <!-- Period type -->
                 <value><i4>3</i4></value>
                 <!-- Trial -->
                 <value><i4>0</i4></value>
                 <!-- Setup Fee -->
                 <value><double>1.000000</double></value>
                 <!-- Subscription Fee -->
                 <value><double>2.000000</double></value>
                 <!-- Renewal Fee -->
                 <value><double>0.000000</double></value>
                 <!-- Transfer Fee -->
                 <value><double>0.000000</double></value>
                 <!-- NonRefundable Amount -->
                 <value><double>0.000000</double></value>
                 <!-- Refund period -->
                 <value><i4>0</i4></value>
                 <!-- Enabled -->
                 <value><i4>1</i4></value>
                 <!-- Number of periods -->
                 <value><i4>12</i4></value>
                 <!-- Fee text -->
                 <value><string></string></value>
                 <!-- Sort number -->
                <value><i4>1</i4></value>
                 <!-- DepositFee -->
                 <value><double>5</double></value>
                 <!-- DepositFeeText -->
                 <value><string></string></value>
               </data>
              </array>
             </value>
            </data>
           </array>
          </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

PlanRateListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns properties of service plan resource rates purchased under at least one of subscriptions. Method returns result only for active, trial, or ordered subscriptions. Method does not return any resource usage statistics.	4	28

Table below illustrates method invocation result for three different cases:

Plan ID	**Subscription ID**	**Subscription status**	**Resource rate ID**	**Result**
1	1000001	Terminated	1,2,5	Empty
	1000002	Terminated	2	
2	none			Empty
3	1000003	Active	1,2,3	Details for resource rates ID 1,2,3, and 4
	1000004	Ordered	3,4	
	1000005	Terminated	3,4,5,6	

Syntax

```
ListResult BM::PlanRateListGet_API(
  Int PlanID;
  Int SortNo;
  Int Reserved1;
  Int Reserved2.
)
returns
  Int ID - resource rate ID;
  Str Name - resource rate name;
  Str Description - resource rate description;
  Double Included Value - resource included value;
  Double Upper Limit - resource upper limit;
  Double Lower Limit - resource lower limit;
  Str Unit of Measure - resource unit of measure;
  Double Setup Fee - resource setup fee;
  Double Recurring Fee - resource recurring fee;
  Int Measurable - signifies if resource is measurable ("1") or not ("0");
  Int ResourceID - resource ID;
  Str SetupFeeDescr - setup fee description, shown in CP and online store;
  Str RecurrFeeDescr - recurring fee description, shown in CP and online store;
  Int IsVisible - signifies if resource rate is shown in CP ("1") or not("0");
  Int IsMain - signifies if resource rate is shown in store ("1") or not("0");;
  Str StoreText - resource description shown in store;
  Int IsSFperUnit - signifies if setup fee is changed per unit ("1") or not ("0");
  Int IsRFperUnit - signifies if recurring fee is changed per unit ("1") or not
("0");
  Int IsSFforUpgrade - signifies if upgrade fee is changed per unit ("1") or not
("0");
  Str StorePriceText - resource price text shown in store (i.e. "5 Gb / month");
  Int ResourceCategoryID - ID of resource category resource is included to;
  Int SortOrder - sort order of resource in store;
  Str Radio Group Name - used for licensed services mostly. If resource has at
least one required and one conflicting resource  (configured under Depends On tab
of resource), two conflicting resources are displayed as option buttons group and
required resource name is taken as the group name;
  Str Parent Resource Name - parent resource name if resource depends on another
resource (common for licensed services);
  Str OveruseFeeDescr - resource overusage fee description;
  Double OveruseFee - resource overusage fee;
  Int IsOveruseFeeTiered - shows whether price tiers are configured for a overuse
fee of the corresponding object (resource rate, resource rate period, resource in
subscription), or not (1) or not (0);
  Int IsRecurringFeeTiered - shows whether price tiers are configured for a
recurring fee of the corresponding object (resource rate, resource rate period,
resource in subscription), or not (1) or not (0).
```

Special Notes

- **Int** PlanID - service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (resource rate ID) in descending order;

- **Int** Reserved1 - reserved parameter, should always be '0';
- **Int** Reserved2 - reserved parameter, should always be '0'.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>PlanRateListGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- PlanID -->
           <value><i4>3</i4></value>
           <!-- SortNo -->
           <value><i4>1</i4></value>
           <!-- Reserved1 -->
           <value><i4>0</i4></value>
           <!-- Reserved2 -->
           <value><i4>0</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
```

```
<data>
 <value>
  <array>
   <data>
    <!-- Resource rate ID -->
    <value><i4>1</i4></value>
    <!-- Resource rate name -->
    <value><string>Diskspace[Unix]</string></value>
    <!-- Resource rate description -->
    <value><string>Linux diskspace</string></value>
    <!-- Included value -->
    <value><double>100.000000</double></value>
    <!-- Upper limit -->
    <value><double>10000000.000000</double></value>
    <!-- Lower limit -->
    <value><double>100.000000</double></value>
    <!-- UOM -->
    <value><string>KB</string></value>
    <!-- Setup fee -->
    <value><double>0.500000</double></value>
    <!-- Recurring fee -->
    <value><double>0.200000</double></value>
    <!-- Measurable -->
    <value><i4>0</i4></value>
    <!-- Resource ID -->
    <value><i4>100001</i4></value>
    <!-- Setup fee description -->
    <value><string>
     Diskspace[Unix] Setup
    </string></value>
    <!-- Recurring fee description -->
    <value><string>
     Diskspace[Unix] Recurring
    </string></value>
    <!-- IsVisible -->
    <value><i4>1</i4></value>
    <!-- IsMain -->
    <value><i4>1</i4></value>
    <!-- Store text -->
    <value><string></string></value>
    <!-- IsSFperUnit -->
    <value><i4>1</i4></value>
    <!-- IsRFperUnit -->
    <value><i4>1</i4></value>
    <!-- IsSFforUpgrade -->
    <value><i4>1</i4></value>
    <!-- StorePriceText -->
    <value><string></string></value>
    <!-- Resource category ID -->
    <value><i4>1</i4></value>
    <!-- SortOrder -->
    <value><i4>999</i4></value>
    <!-- Radio group name -->
    <value><string></string></value>
    <!-- Parent resource name -->
    <value><string>Diskspace</string></value>
            <!-- Overuse Fee Description -->
    <value><string>Diskspace overusage</string></value>
```

```
                      <!-- Overuse Fee -->
                 <value><double>440.000000</double></value>
                 <!-- IsOveruseFeeTiered -->
                 <value><i4>0</i4></value>
                 <!-- IsRecurringFeeTiered -->
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanRateListGetForPeriod_API

Description	Parameters	
	Passed	Returned
Method returns resource rates properties of specified plan subscription period. Method returns resources rates properties only for resources included into a resource category and shown in online store.	2	27

Syntax

```
ListResult BM::PlanRateListGetForPeriod_API(
  Int PlanPeriodID;
  Int SortNo.
)
returns
  Int ID - resource rate ID;
  Str Name - resource rate name;
  Str Description - resource rate description;
  Double Included Value - resource included value;
  Double Upper Limit - resource upper limit;
  Double Lower Limit - resource lower limit;
  Str Unit of Measure - resource unit of measure;
  Double Setup Fee - resource setup fee;
  Double Recurring Fee - resource recurring fee;
  Int Measurable - signifies if resource is measurable ("1") or not ("0");
  Int ResourceID - resource ID;
  Str SetupFeeDescr - setup fee description, shown in CP and online store;
  Str RecurrFeeDescr - recurring fee description, shown in CP and online store;
  Int IsVisible - signifies if resource rate is shown in CP ("1") or not("0");
  Int IsMain - signifies if resource rate is shown in store ("1") or not("0");;
  Str StoreText - resource description shown in store;
  Int IsSFperUnit - signifies if setup fee is changed per unit ("1") or not ("0");
  Int IsRFperUnit - signifies if recurring fee is changed per unit ("1") or not
("0");
  Int IsSFforUpgrade - signifies if upgrade fee is changed per unit ("1") or not
("0");
  Str StorePriceText - resource price text shown in store (i.e. "5 Gb / month");
  Int ResourceCategoryID - ID of resource category resource is included to;
  Int SortOrder - sort order of resource in store;
  Str Radio Group Name - used for licensed services mostly. If resource has at
least one required and one conflicting resource  (configured under Depends On tab
of resource), two conflicting resources are displayed as option buttons group and
required resource name is taken as the group name;
  Str Parent Resource Name - parent resource name. If resource depends on another
resource (common for licensed services);
  Int Parent Depend Multiplier - defines multiplier of dependant resource. For
example, 1 mailbox requires 1024 KB of diskspace, so 5 mailboxes require 5120 KB,
1024 is multiplier;
  Int IsOveruseFeeTiered - shows whether price tiers are configured for a overuse
fee of the corresponding object (resource rate, resource rate period, resource in
subscription), or not (1) or not (0);
```

```
  Int IsRecurringFeeTiered - shows whether price tiers are configured for a
recurring fee of the corresponding object (resource rate, resource rate period,
resource in subscription), or not (1) or not (0).
```

Special Notes

- **Int** PlanPeriodID - subscription period ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (resource rate ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>PlanRateListGetForPeriod_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- PlanPeriodID -->
           <value><i4>3</i4></value>
           <!-- SortNo -->
           <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Resource rate ID -->
                 <value><i4>1</i4></value>
                 <!-- Resource rate name -->
                 <value><string>
                  Diskspace[Unix]
                 </string></value>
                 <!-- Resource rate description -->
                 <value><string>
                  Linux diskspace
                 </string></value>
                 <!-- Included value -->
                 <value><double>100.000000</double></value>
                 <!-- Upper limit -->
                 <value><double>10000000.000000</double></value>
                 <!-- Lower limit -->
                 <value><double>100.000000</double></value>
                 <!-- UOM -->
                 <value><string>KB</string></value>
                 <!-- Setup fee -->
                 <value><double>0.500000</double></value>
                 <!-- Recurring fee -->
                 <value><double>0.200000</double></value>
                 <!-- Measurable -->
                 <value><i4>0</i4></value>
                 <!-- Resource ID -->
                 <value><i4>100001</i4></value>
                 <!-- Setup fee description -->
                 <value><string>Diskspace[Unix] Setup</string></value>
                 <!-- Recurring fee description -->
                 <value><string>Diskspace[Unix] Recurring</string></value>
                 <!-- IsVisible -->
                 <value><i4>1</i4></value>
                 <!-- IsMain -->
                 <value><i4>1</i4></value>
                 <!-- Store text -->
                 <value><string></string></value>
                 <!-- IsSFperUnit -->
                 <value><i4>1</i4></value>
                 <!-- IsRFperUnit -->
                 <value><i4>1</i4></value>
                 <!-- IsSFforUpgrade -->
                 <value><i4>1</i4></value>
                 <!-- StorePriceText -->
                 <value><string></string></value>
                 <!-- Resource category ID -->
                 <value><i4>1</i4></value>
                 <!-- SortOrder -->
                 <value><i4>999</i4></value>
                 <!-- Radio group name -->
                 <value><string></string></value>
                 <!-- Parent resource name -->
                 <value><string>Diskspace</string></value>
                 <!-- Parent Depend Multiplier -->
                 <value><i4>10</i4></value>
                 <!-- IsOveruseFeeTiered -->
                 <value><i4>0</i4></value>
                 <!-- IsRecurringFeeTiered -->
                 <value><i4>0</i4></value>
                </data>
```

```
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

PlanRateListFullGet_API

Description	Parameters	
	Passed	Returned
Method returns resource rates properties of specified service plan. Method returns resources rates properties only for resources included into a resource category.	2	24

Syntax

```
ListResult BM::PlanRateListFullGet_API(
  Int PlanID;
  Int SortNo.
)
returns
  Int ID - resource rate ID;
  Str Name - resource rate name;
  Str Description - resource rate description;
  Double Included Value - resource included value;
  Double Upper Limit - resource upper limit;
  Double Lower Limit - resource lower limit;
  Str Unit of Measure - resource unit of measure;
  Double Setup Fee - resource setup fee;
  Double Recurring Fee - resource recurring fee;
  Int Measurable - signifies if resource is measurable ("1") or not ("0");
  Int ResourceID - resource ID;
  Str SetupFeeDescr - setup fee description, shown in CP and online store;
  Str RecurrFeeDescr - recurring fee description, shown in CP and online store;
  Int IsVisible - signifies if resource rate is shown in CP ("1") or not("0");
  Int IsMain - signifies if resource rate is shown in store ("1") or not("0");;
  Str StoreText - resource description shown in store;
  Int IsSFperUnit - signifies if setup fee is changed per unit ("1") or not ("0");
  Int IsRFperUnit - signifies if recurring fee is changed per unit ("1") or not
("0");
  Int IsSFforUpgrade - signifies if upgrade fee is changed per unit ("1") or not
("0");
  Str StorePriceText - resource price text shown in store (that is, "5 Gb /
month");
  Int ResourceCategoryID - ID of resource category resource is included to;
  Int SortOrder - sort order of resource in store;
  Int IsOveruseFeeTiered - shows whether price tiers are configured for a overuse
fee of the corresponding object (resource rate, resource rate period, resource in
subscription), or not (1) or not (0);
  Int IsRecurringFeeTiered - shows whether price tiers are configured for a
recurring fee of the corresponding object (resource rate, resource rate period,
resource in subscription), or not (1) or not (0).
```

Special Notes

- **Int** PlanID - service plan ID;

- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (resource rate ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>PlanRateListFullGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- PlanID -->
           <value><i4>3</i4></value>
          <!-- SortNo -->
           <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Resource rate ID -->
                 <value><i4>1</i4></value>
                 <!-- Resource rate name -->
                 <value><string>Diskspace[Unix]</string></value>
                 <!-- Resource rate description -->
                 <value><string>Linux diskspace</string></value>
                 <!-- Included value -->
                 <value><double>100.000000</double></value>
                 <!-- Upper limit -->
                 <value><double>10000000.000000</double></value>
                <!-- Lower limit -->
                 <value><double>100.000000</double></value>
                <!-- UOM -->
                 <value><string>KB</string></value>
                 <!-- Setup fee -->
                 <value><double>0.500000</double></value>
                 <!-- Recurring fee -->
                 <value><double>0.200000</double></value>
                 <!-- Measurable -->
                 <value><i4>0</i4></value>
                 <!-- Resource ID -->
                 <value><i4>100001</i4></value>
                 <!-- Setup fee description -->
                 <value><string>Diskspace[Unix] Setup</string></value>
                 <!-- Recurring fee description -->
                 <value><string>Diskspace[Unix] Recurring</string></value>
                 <!-- IsVisible -->
                 <value><i4>1</i4></value>
                 <!-- IsMain -->
                 <value><i4>1</i4></value>
                 <!-- Store text -->
                 <value><string> </string></value>
                 <!-- IsSFperUnit -->
                 <value><i4>1</i4></value>
                 <!-- IsRFperUnit -->
                 <value><i4>1</i4></value>
                 <!-- IsSFforUpgrade -->
                 <value><i4>1</i4></value>
                 <!-- StorePriceText -->
                 <value><string> </string></value>
                 <!-- Resource category ID -->
                 <value><i4>1</i4></value>
                 <!-- SortOrder -->
                 <value><i4>999</i4></value>
                 <!-- IsOveruseFeeTiered -->
                 <value><i4>0</i4></value>
                 <!-- IsRecurringFeeTiered -->
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
```

```
    </param>
  </params>
</methodResponse>
```

PlanResourceListGet_API

Description	Parameters	
	Passed	**Returned**
Methods returns properties of service plan resource rates for resources that are included in particular resource category. Result can be truncated depending on resource rate **Included Units** parameter value.	4	17

Syntax

```
ListResult BM::PlanResourceListGet_API(
  Int PlanID;
  Int ResourceCategoryID;
  Int StatusList;
  Int SortNo.
)
returns
  Int ResourceID - resource ID;
  Str(250) ResourceName - resource name;
  Str(1024) Description - resource rate description;
  Double SetupFee - resource rate setup fee;
  Double RecurringFee - resource rate recurring fee;
  Double CostForAdditional resource rate overuse fee;
  Double IncludedValue - resource included units;
  Double MaxValue - resource upper limit;
  Double MinValue - resource lower limit;
  Int IsMain - signifies if resource rate is shown in store ("1") or not("0");
  Int IsSFperUnit - signifies if setup fee is changed per unit ("1") or not ("0");
  Int IsVisible - signifies if resource rate is shown in CP ("1") or not("0");
  Str(30) MeasureUnit - resource unit of measure;
  Int ResourceCategoryID - resource category ID;
  Int SortOrder - sort order of resource in store;
  Int IsOveruseFeeTiered - shows whether price tiers are configured for a overuse
fee of the corresponding object (resource rate, resource rate period, resource in
subscription), or not (1) or not (0);
  Int IsRecurringFeeTiered - shows whether price tiers are configured for a
recurring fee of the corresponding object (resource rate, resource rate period,
resource in subscription), or not (1) or not (0).
```

Special Notes

- **Int** PlanID - service plan ID;
- **Int** ResourceCategoryID - resource category ID;
- **Int** StatusList - truncates result depending on resource rate **Included Unit** value:
  - StatusList == 0. Method returns resources, which have positive **Included Unit** value in resource rate (Caution! Resource rate != resource in service template)

    - StatusList == 1. Method returns resources, which have zero **Included Unit** value in resource rate (Caution! Resource rate != resource in service template)
    - StatusList empty. Method returns all resources of plan resource rates, which belong to specified resource category.
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Resource ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>PlanResourceListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>3</i4></value>
        <!-- ResourceCategoryID -->
         <value><i4>1</i4></value>
         <!-- StatusList -->
         <value><i4></i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Resource ID -->
                 <value><i4>100001</i4></value>
                 <!-- Resource name -->
                 <value><string>Diskspace[Unix]</string></value>
                 <!-- Resource description -->
                 <value><string>Linux diskspace</string></value>
                 <!-- Setup fee -->
                 <value><double>0.500000</double></value>
                 <!-- Recurring fee -->
                 <value><double>0.200000</double></value>
                 <!-- Overuse fee -->
                 <value><double>0.200000</double></value>
                 <!-- Included value -->
                 <value><double>100.000000</double></value>
                 <!-- Upper limit -->
                 <value><double>10000000.000000</double></value>
                 <!-- Lower limit -->
                 <value><double>100.000000</double></value>
                 <!-- IsMain -->
                 <value><i4>1</i4></value>
                 <!-- IsSFperUnit -->
                 <value><i4>1</i4></value>
                 <!-- IsVisible -->
                 <value><i4>1</i4></value>
                 <!-- UOM -->
                 <value><string>KB</string></value>
                 <!-- Resource category ID -->
                 <value><i4>1</i4></value>
                 <!-- SortOrder -->
                 <value><i4>999</i4></value>
                 <!-- IsOveruseFeeTiered -->
                 <value><i4>0</i4></value>
                 <!-- IsRecurringFeeTiered -->
                 <value><i4>0</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

RenewDomainListGet_API

	Parameters	
Description	**Passed**	**Returned**
Methods returns the list of domains in the *Registered* status and with non-zero expiration date for the specified account.	2	6

Syntax

```
ListResult DOMAINGATE::RenewDomainListGet_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int DomainID - domain ID;
  Int SubscriptionID - domain subscription ID;
  Str Domain - domain name;
  Int Status - domain status, always returns "20" - registered;
  Int ExpirationDate - domain expiration date in Unix date format;
  Int PlanID - service plan ID.
```

Special Notes

- **Int** AccountID - customer account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>RenewDomainListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000001</i4></value>
         <!-- Sort No -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- DomainID -->
              <value><i4>1</i4></value>

              <!-- SubscriptionID -->
              <value><i4>1000001</i4></value>

              <!-- Domain name-->
              <value><string>domainname.com</string></value>

              <!--Domain status  -->
              <value><i4>20</i4></value>

              <!-- Expiration Date -->
              <value><i4>1302175122</i4></value>

              <!-- Service Plan ID -->
              <value><i4>2</i4></value>
             </data>
            </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

ResourceCategoryDetailsGet_API

Description	Parameters	
	Passed	**Returned**
Method returns details for specified resource category.	2	9

Syntax

```
ItemResult BM::ResourceCategoryDetailsGet_API(
  Int ResCatID;
  Int AccountID.
)
returns
  Int ResCatID - requested resources category ID;
  Str(30) Name - requested resource category name;
  Str(1024) Description - requested resource category description;
  Int ShopDisplayType - defines how resources are organized within resource
category in online store: "0" - resources are shown as a group of check boxes, a
number of resources can be purchased at a time; "1" - resources are shown as
option buttons group, one resource can be purchased at a time;
  Int SortNumber - serial number of resource category in online store (returns '-
2', if not specified).
  Int Parent_ResCatID - parent resource category ID (returns '-2', if not
specified);
  Int Default_ResourceID - default resource ID (returns '-2', if not specified).
Default resource is selected by default in online store. Applicable for resources
categories with ShopDisplayType=1;
  Int Default_ResCatID - default child resource category ID (returns '-2', if not
specified). Default child resource category is selected by default in online
store. Applicable for resources categories with ShopDisplayType=1;
  Int IsOptional - signifies whether "None" value is added to the resources list
of the category in online store: "0" - "None" is absent, "1" - "None" is added.
Applicable for resources categories with ShopDisplayType=1.
```

Special Notes

- **Int** ResCatID - resource category ID;
- **Int** AccountID - vendor account ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceCategoryDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ResCatID -->
         <value><i4>10140</i4></value>
         <!-- AccountID -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>

              <!-- Resource Category ID -->
              <value><i4>10140</i4></value>

              <!-- Resource Category name -->
              <value><string>Language Packs</string></value>

              <!-- Resource Category description -->
              <value><string>Language Packs</string></value>

              <!-- ShopDisplayType -->
              <value><i4>1</i4></value>

              <!-- SortNumber -->
              <value><i4>10140</i4></value>

              <!-- Parent Resource Category ID -->
              <value><i4>10000</i4></value>

              <!-- Default  ResourceID -->
              <value><i4>-2</i4></value>

              <!-- Default  ResCatID -->
              <value><i4>-2</i4></value>

              <!-- IsOptional -->
              <value><i4>1</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ResourceCategoryByPlanListGet_API

Description	Parameters	
	Passed	Returned
Method returns list of resource categories related to specified service plan. Service plan resources rates can be configured for resources that belongs to different resource categories.	3	9

Syntax

```
ListResult BM::ResourceCategoryByPlanListGet_API(
  Int PlanID;
  Int SortNo;
)
returns
  Int ResCatID - resources category ID;
  Str(30) Name - resource category name;
  Str(1024) Description - resource category description;
  Int ShopDisplayType - defines how resources are organized within resource
category in online store: "0" - resources are shown as a group of check boxes, a
number of resources can be purchased at a time; "1" - resources are shown as
option buttons group, one resource can be purchased at a time;
  Int SortNumber - serial number of resource category in online store (returns '-
2', if not specified).
  Int Parent_ResCatID - parent resource category ID (returns '-2', if not
specified);
  Int Default_ResourceID - default resource ID (returns '-2', if not specified).
Default resource is selected by default in online store. Applicable for resources
categories with ShopDisplayType=1.
  Int Default_ResCatID - default child resource category ID (returns '-2', if not
specified). Default child resource category is selected by default in online
store. Applicable for resources categories with ShopDisplayType=1.
  Int IsOptional - signifies whether "None" value is added to the resources list
of the category in online store: "0" - "None" is absent, "1" - "None" is added.
Applicable for resources categories with ShopDisplayType=1.
```

Special Notes

- **Int** PlanID - service plan ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 2 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (ResCat ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceCategoryByPlanListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanID -->
         <value><i4>18</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Resource Category ID -->
              <value><i4>10140</i4></value>
              <!-- Resource Category name -->
              <value><string>Language Packs</string></value>
              <!-- Resource Category description -->
              <value><string>Language Packs</string></value>
              <!-- ShopDisplayType -->
              <value><i4>1</i4></value>
              <!-- SortNumber -->
              <value><i4>10140</i4></value>
              <!-- Parent Resource Category ID -->
              <value><i4>10000</i4></value>
              <!-- Default  ResourceID -->
              <value><i4>-2</i4></value>
              <!-- Default  ResCatID -->
              <value><i4>-2</i4></value>
              <!-- IsOptional -->
              <value><i4>1</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ResourceDependenceSet_API

Description	Parameters	
	Passed	**Returned**
Method adds, updates or removes resources dependencies from database.	5	1

Syntax

```
ErrorMessage BM::ResourceDependenceSet_API(
  Int ResourceID;
  Int ParentResID;
  Int DependencyType;
  Int AddRemove;
  Int DepMultiplier.
)
returns
  Str ErrorMessage - message after resource dependencies has been deleted, added or
updated: "Operation done."
```

Special Notes

- **Int** ResourceID - dependant resource ID;
- **Int** ParentResID - parent resource name ID;
- **Int** DependencyType - dependency type: 1 - resource requires parent one for installation, 2 - resource conflicts with parent one, 3 - resource is provided by parent one;
- **Int** AddRemove - operation to perform: 0 - remove dependency, 1 - add or update existing dependency;
- **Int** DepMultiplier - defines multiplier of dependant resource. For example, 1 mailbox requires 1024 KB of diskspace, so 5 mailboxes require 5120 KB, 1024 is multiplier. If multiplier is not needed pass empty.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceDependenceSet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ResourceID -->
         <value><i4>13555</i4></value>
         <!-- ParentResID -->
         <value><i4>65271</i4></value>
         <!-- DependencyType -->
         <value><i4>2</i4></value>
         <!-- AddRemove -->
         <value><i4>0</i4></value>
         <!-- DepMultiplier -->
         <value><i4></i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
```

```
            <value>
             <struct>
              <member>
               <name>Status</name>
                <value><string>
                 Operation done.
               </string></value>
              </member>
             </struct>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
  </methodResponse>
```

ResourceFromTemplateRemove_API

	Parameters	
Description	**Passed**	**Returned**
Method removes link between specified resource and service template. Resource is removed from subscriptions and service plan resource rates also. If resource was upgraded in some subscriptions, you should set its value to **Included** value defined in service plan and then remove the link. **Note:** ResourceFromTempalteRemove_API is obsolete and left for backward compatibility.	2	1

Syntax

```
ErrorMessage BM::ResourceFromTemplateRemove_API(
  Int ServiceTemplateID;
  Int ResourceID.
)
returns
  Str ErrorMessage -  message after link has been removed: "Data have been
deleted."
```

Special Notes

- **Int** ServiceTemplateID - service template ID;
- **Int** ResourceID - resource ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceFromTemplateRemove_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ServiceTemplateID -->
         <value><i4>1000003</i4></value>
         <!-- ResourceID -->
         <value><i4>100003</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
```

```
               <value><string>
                Data have been deleted.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ResourceListByCategoryGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of resources that belongs to selected resource category.	3	5

Syntax

```
ListResult BM::ResourceListByCategoryGet_API(
  Int ResCatID;
  Int AccountID;
  Int SortNo.
)
returns
  Int ResourceID - resource ID;
  Str(250) Name - resource name;
  Str(1024) Description - resource description;
  Str(30) MeasureUnit - resource UOM;
  Int ResCatID - requested resource category ID.
```

Special Notes

- **Int** ResCatID - resource category ID;
- **Int** AccountID - vendor account ID (provider or reseller);
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 2 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Resource ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceListByCategoryGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ResCatID -->
         <value><i4>1</i4></value>
         <!-- AccountID -->
         <value><i4>1</i4></value>
        <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
```

```
             <data>
              <value>
               <array>
                <data>
                 <!-- Resource ID -->
                 <value><i4>100001</i4></value>
                 <!-- Resource Name -->
                 <value><string>Diskspace[Unix]</string></value>
                 <!-- Resource Description -->
                 <value><string>Linux diskspace</string></value>
                 <!-- Resource UOM -->
                 <value><string>KB</string></value>
                 <!-- Resource Category ID -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
              <value>
               <array>
                <data>
                 <!-- Resource ID -->
                 <value><i4>100002</i4></value>
                 <!-- Resource Name -->
                 <value><string>Traffic[Unix]</string></value>
                 <!-- Resource Description -->
                 <value><string>Linux Traffic</string></value>
                 <!-- Resource UOM -->
                 <value><string>KB</string></value>
                 <!-- Resource Category ID -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ResourceRemove_API

Description	Parameters	
	Passed	Returned
Method removes resource from database. Resource can not be removed if it is used in a service template or service plan resource rate or has other dependencies.	1	1

Syntax

```
ErrorMessage BM::ResourceRemove_API(
  Int ResourceID.
)
returns
  Str ErrorMessage - message after resource has been deleted: "Resource # has been
removed."
```

Special Notes

Int ResourceID - resource ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceRemove_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- ResourceID -->
         <value><i4>100003</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
               Resource #100003 has been removed.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ResourceSync_API

Envelope 1

Description	Parameters	
	Passed	Returned
Method updates resource with provided parameters in PBA database. If resource does not exist, it is added to database and bounded to specified service template. Also method is used for updating resources from POA (versions earlier then 2.8).	11	1

Syntax

```
ErrorMessage BM::ResourceSync_API(
  Int STID;
  Int ResourceID;
  Str(250) Name;
  Str(1024) Description;
  Int Additive;
  Int Measurable;
  Str(30) MeasureUnit;
  Int MinValue;
  Int MaxValue;
  Int Multiplier;
  Str(30) Command.
)
returns
  Str ErrorMessage - message after resource has been updated: "Operation done."
```

Special Notes

- **Int** ServiceTemplateID - ID of existing service template;
- **Int** ResourceID - ID of existing or new resource;
- **Str(250)** Name - new resource name;
- **Str(1024)** Description - new resource description;
- **Int** Additive - defines whether resource is additive: '0' - not additive, '1' - additive;
- **Int** Measurable - defines whether resource is measurable: '0' - not measurable, '1' - measurable, '-2' - undefined. If the '-2' value is passed for a new resource, the resource becomes measurable. If the '-2' value is passed for an existing resource, the resource **Measurable** parameter value remains unchanged;
- **Str(30)** MeasureUnit - resource UOM registered in PBA;

- **Int** MinValue - minimal available value of the resource for service template;
- **Int** MaxValue - maximum available value of the resource for service template;
- **Int** Multiplier - resource multiplier in the external system. For example, 1024 - to send kilobytes if the resource base unit is megabytes and the external system, for example Virtuozzo requires kilobytes;
- **Str(30)** Command - defines the command to configure the resource. The parameter is optional and used for specific service providers only.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceSync_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- STID -->
         <value><i4>123</i4></value>
         <!-- ResourceID -->
         <value><i4>123</i4></value>
         <!-- Name -->
         <value>Diskspace</value>
         <!-- Description -->
         <value>Diskspace for Linux</value>
         <!-- Additive -->
         <value><i4>0</i4></value>
        <!-- Measurable -->
         <value><i4>1</i4></value>
         <!-- MeasureUnit -->
         <value>KB</value>
         <!-- minValue -->
         <value><i4>100</i4></value>
         <!-- maxValue -->
         <value><i4>1000</i4></value>
         <!-- Multiplier -->
         <value><i4>1024</i4></value>
         <!-- Command -->
         <value></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

Envelope 2

Description	Parameters	
	Passed	Returned
Method updates resource with provided by POA (version 2.8 and higher) parameters in PBA database. If resource does not exist, it is added to database and bounded to specified service template.	12..N	1

Syntax

```
ErrorMessage PEMGATE::ResourceSync_API(
  Int STID;
  Int ResourceID;
  Str(250) Name;
  Int ClassID
  Str(1024) Description;
  Int Additive;
  Int Measurable;
  Str(30) MeasureUnit;
  Int MinValue;
  Int MaxValue;
  Int Multiplier;
  Str(30) Command;
  Int iC2UParams;
  Str Key_1;
  Str Value_1;
  ...
  Str Key_N;
  Str Value_N;
  Int iRevenueParams;
  Str Key_1;
  Str Value_1;
  ...
  Str Key_N;
  Str Value_N.
)
returns
  Str ErrorMessage - message after resource has been updated: "Operation done."
```

Special Notes

- **Int** ServiceTemplateID - ID of existing service template;
- **Int** ResourceID - ID of existing or new resource;
- **Str(250)** Name - new resource name;
- **Int** ClassID - POA resource class ID;
- **Str(1024)** Description - new resource description;
- **Int** Additive - defines whether resource is additive: 0 - not additive, 1 - additive;

- **Int** Measurable - defines whether resource is measurable: '0' - not measurable, '1' - measurable, '-2' - undefined. If the '-2' value is passed for a new resource, the resource becomes measurable. If the '-2' value is passed for an existing resource, the resource **Measurable** parameter value remains unchanged;
- **Str(30)** MeasureUnit - resource UOM registered in PBA;
- **Int** MinValue - minimal available value of the resource for service template;
- **Int** MaxValue - maximum available value of the resource for service template;
- **Int** Multiplier - resource multiplier in the external system. For example, 1024 - to send kilobytes if the resource base unit is megabytes and the external system, for example Virtuozzo requires kilobytes;
- **Str(30)** Command - defines the command to configure the resource. The parameter is optional and used for specific service providers only.
-
- **Optional parameters**
- **Int** iC2UParams - c2u parameters count; the number of pairs (parameter and value) following above;
- **Str** Key_1, **Str** Value_1, ..., **Str** Key_N, **Str** Value_N - c2u parameters pairs;
- **Int** iRevenueParams - revenue parameters count; the number of pairs (parameter and value) following right after;
- **Str** Key_1, **Str** Value_1, ..., **Str** Key_N, **Str** Value_N - revenue parameters pairs.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>PEMGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>ResourceSync_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- STID -->
         <value><i4>123</i4></value>
         <!-- ResourceID -->
         <value><i4>123</i4></value>
         <!-- Name -->
         <value>Diskspace</value>
         <!--ClassID -->
         <value><i4>32</i4></value>
         <!-- Description -->
         <value>Diskspace for Linux</value>
         <!-- Additive -->
         <value><i4>0</i4></value>
         <!-- Measurable -->
         <value><i4>1</i4></value>
         <!-- MeasureUnit -->
         <value>KB</value>
         <!-- minValue -->
         <value><i4>100</i4></value>
         <!-- maxValue -->
         <value><i4>1000</i4></value>
         <!-- Multiplier -->
         <value><i4>1024</i4></value>
         <!-- Command -->
         <value></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SalesBranchAccountSet_API

Description	Parameters	
	Passed	**Returned**
Method adds a sales branch to the specified account.	2	1

Syntax

```
BM::SalesBranchAccountSet_API(
  Int AccountID;
  Str BranchID.
)
returns
  Str Message.
```

Parameters description

Input parameters:

- **Int** AccountID - identifier of the user account which is to be associated with the sales branch.
- **Str** BranchID - identifier of the sales branch.

Output parameter:

- **Str** Message - message issued when the sales branch is added to the specified account. The message content is as follows: `Your changes have been saved.`

Example

Request

```
SalesBranchAccountSet_API:
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>SalesBranchAccountSet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
       <!-- AccountID -->
          <value><int>1000270</int></value>
      <!-- BranchID -->
          <value><string>1</string></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <struct>
            <member>
             <name>Status</name>
              <value>
               <string>Your changes have been saved. </string>
              </value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SalesBranchAdd_API

	Parameters	
Description	**Passed**	**Returned**
Method adds a new sales branch.	3	1

Syntax

```
BM::SalesBranchAdd_API(
  Str BranchID;
  Int VendorID;
  Str(80) Name.
)
returns
  Str Message.
```

Parameters description

Input parameters:

- **Str** BranchID - Identifier of the new sales branch. This parameter is optional.
- **Int** VendorID - Identifier of the vendor associated with the created sales branch.
- **Str(80)** Name - Name of the new sales branch. This parameter is optional.

Output parameter:

- **Str** Message - Message issued when the sales branch is added. The message content is as follows: `Data have been successfully added.`

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>SalesBranchAdd_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- BranchID-this parameter is optional-->
           <value><string>1</string></value>
           <!-- VendorID-this parameter is required-->
           <value><i4>1</i4></value>
           <!-- Name-this parameter is optional-->
           <value<string>TestBranch</string></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
              <value>
               <string>Data have been successfully added.</string>
              </value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SalesCategoryListGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of active sales categories for specified vendor account.	1	7

Syntax

```
ListResult BM::SalesCategoryListGet_API(
  Int AccountID;
  Int SortNo.
)
returns
  Int CategoryID - sales category ID;
  Str(30) Name - sales category name;
  Str(1024) ShortDescription - sales category short description;
  Str(1024) LongDescription - sales category full description;
  Int DisplayByDefault - defines how category service plans are displayed in store:
"1" - with description expanded by default,  "0" - with description hidden by
default;
  Int DocID - internal ID of category icon (displayed in online store);
  Str(16384) EulaURL - URL to EULA page.
```

Special Notes

- **Int** AccountID - vendor account ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 2 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Category ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SalesCategoryListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>2</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>
                 <!-- Sales Category name -->
                 <value><string>Domain Registration</string></value>
                 <!-- Sales Category description -->
                 <value><string>Domain registration category</string></value>
                 <!-- Sales Category long description -->
                 <value><string> </string></value>
                 <!-- DisplayByDefault -->
                 <value><i4>0</i4></value>
                 <!-- Icon ID -->
                 <value><i4>1023</i4></value>
                 <!-- URL to EULA page -->
                 <value><string>
          I have read and I accept the
          <a href="example.com/customer-agreement.html">
          Provider, Inc. Terms of Service</a>.
          I also understand that Domain Name registration
          orders cannot assure availability and are subject
          to these same Terms and Conditions.
                 </string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SalesCategoryDetailsGet_API

Description	Parameters	
	Passed	**Returned**
Method returns details for specified sales category.	1	10

Syntax

```
ItemResult BM::SalesCategoryDetailsGet_API(
  Int CategoryID.
)
returns
  Int CategoryID - sales category ID;
  Str(30) Name - sales category name;
  Str(1024) ShortDescription - sales category short description;
  Int AccountID - ID of vendor account the category belongs to;
  Str(1024) LongDescription - sales category long description;
  Int DisplayByDefault - defines how category service plans are displayed in store:
"1" - with description expanded by default,  "0" - with description hidden by
default;
  Int DocID - internal ID of category icon (displayed in online store);
  Str(16384) EulaURL - URL od EULA page;
  Str ImageName - internal name of category icon: icon ID + file extension.
```

Special Notes

Int CategoryID - sales category ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SalesCategoryDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- CategoryID -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Sales Category ID -->
                 <value><i4>1</i4></value>

                 <!-- Sales Category name -->
                 <value><string>Domain Registration</string></value>

                 <!-- Sales Category description -->
                 <value><string>Domain registration category</string></value>

                 <!-- Vendor account ID -->
                 <value><i4>1</i4></value>

                 <!-- Sales Category long description -->
                 <value><string> </string></value>

                 <!-- DisplayByDefault -->
                 <value><i4>0</i4></value>

                 <!-- Icon ID -->
                 <value><i4>1023</i4></value>

                 <!-- URL to EULA page -->
                 <value><string>
I have read and I accept the
 <a href="example.com/customer-agreement.html">
 Provider, Inc. Terms of Service</a>.
 I also understand that Domain Name registration orders
 cannot assure availability
 and are subject to these same Terms and Conditions.
                 </string></value>

                 <!-- Image name -->
                 <value>1023.gif </value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SalesPersonAccountSet_API

	Parameters	
Description	**Passed**	**Returned**
The method assigns a new sales person for the specified account.	2	1

Syntax

```
ErrorMessage BM::SalesPersonAccountSet_API(
  Int AccountID;
  Str SalesID.
  )
returns
  Str ErrorMessage.
```

Parameters Description

Input Parameters:

- **Int** AccounID – Identifier of an account, which requires changing the sales person.
- **Str** SalesID – Identifier of a new sales person that is to be assigned to the account.

Output Parameter:

Str ErrorMessage – Message issued after updating the SalesID parameter value. The message content is as follows: "Your changes have been saved".'

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SalesPersonAccountSet_API</value>
     </member>
     <member>
```

```
      <name>Params</name>
      <value>
       <array>
        <data>
                                   <!-- AccountID -->
               <value><i4>1000008</i4></value>
                            <!-- SalesID -->
         <value><string>5</string></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Your changes have been saved.</string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

SalesPersonAdd_API

Description	Parameters	
	Passed	Returned
Method creates a new sales person.	6	1

Syntax

```
ItemResult BM::SalesPersonAdd_API(
  Str SalesID;
  Int VendorID;
  Str(80) Name;
  Str BranchID;
  Double SalesCommission;
  Double RecurringCommission.
)
returns
  Str Message.
```

Parameters description

Input parameters:

- **Str** SalesID - Identifier of a sales person that you are going to add. This parameter is optional.
- **Int** VendorID - Vendor for which you want to create the sales person.
- **Str(80)** Name - Name of the sales person that you are going to create. This parameter is optional.
- **Str** BranchID - Identifier of the sales branch associated with the specified sales person. This parameter is optional.
- **Double** SalesCommission - Sales person commission for sales (percentagewise). Sales commission is a percent of setup fee specified in invoice
- **Double** RecurringCommission - Sales person commission for recurring payments (percentagewise). Recurring commission is a percent of recurring fee specified in invoice.

Output parameter:

- **Str** Message - message issued when the new sales person is added. The message content is as follows: `Data have been successfully added.`

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse the XML comments, REMOVE all the comments from an actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>SalesPersonAdd_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- SalesID - This parameter is optional-->
           <value><string>1</string></value>
           <!-- VendorID - This parameter is required-->
           <value><i4>1</i4></value>
           <!-- Name - This parameter is optional -->
           <value><string>TestPerson</string></value>
           <!-- BranchID - This parameter is optional-->
           <value><string>1</string></value>
           <!-- SalesCommission - This parameter is required-->
           <value>1</value>
           <!-- RecurringCommission - This parameter is required-->
           <value>1</value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
              <value>
               <string>Data have been successfully added.</string>
              </value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SendSubscriptionNotificationForUser_API

Description	Parameters	
	Passed	**Returned**
Method is used to send a notification with set a set of customer placeholders	3 or more	1

Syntax

```
ErrorMessage * MESSAGE::SendSubscriptionNotificationForUser_API(
  Str TemplateName;
  Int SubscriptionID;
  Int UserID;
  Str Key_1
  Str Value_1
  ...
  Str Key_N
  Str Value_N.
  )
returns
 Str ErrorMessage - message after sending notification: "Operation done". Common
messages on failing to send: "Template is not found. Can not send notification.",
"Access denied.", "Incorrect packet format.".
```

Special Notes

Str TemplateName - name of the notification template;

Int SubscriptionID - subscription ID;

Int UserID - ID of the user to send notification to;

Str Key_N, **Str** Value_N - an optional set of pairs (parameter and value) to be included to notification. The parameters and values represent placeholder name and its value respectively.

Example

Request

```
<?xml version="1.0"?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
          <member>
```

```
           <name>Server</name>
            <value>MESSAGE</value>
          </member>
          <member>
           <name>Method</name>
            <value>SendSubscriptionNotificationForUser_API</value>
          </member>
          <member>
           <name>Params</name>
            <value>
              <array>
               <data>
                <value>
                 <!-- TemplateName -->
                 <string>Welcome e-mail - Dedicated Hosting</string>
                </value>
                <value>
                 <!-- SubscriptionID -->
                 <i4>1000220</i4>
                </value>
                <value>
                 <!-- UserID -->
                 <i4>1000016</i4>
                </value>
                <value>
                 <!-- Key  1 -->
                 <string>XXX</string>
                </value>
                <value>
                 <!-- Value  1 -->
                 <string>YYY</string>
                </value>
               </data>
              </array>
            </value>
           </member>
          </struct>
         </value>
        </param>
       </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
          <name>Result</name>
           <value>
            <array>
              <data>
               <value>
                <struct>
                 <member>
                  <name>Status</name>
                 <value>
                  <string>Operation done.</string>
                 </value>
                </member>
               </struct>
              </value>
            </data>
           </array>
          </value>
         </member>
        </struct>
       </value>
  </param>
 </params>
</methodResponse>
```

ServiceTemplateRemove_API

Description	Parameters	
	Passed	Returned
Method removes service template from database. Service template can not be removed if there are service plans and subscriptions created on its base. **Important!** The method should be used for templates created in PBA only. It is forbidden to delete service templates created by any external system, for example POA.	1	1

Syntax

```
ErrorMessage BM::ServiceTemplateRemove_API(
  Int ServiceTemplateID.
)
returns
  Str ErrorMessage - message after service template has been deleted: "All selected
Service Templates have been removed."
```

Special Notes

Int ServiceTemplateID - service template ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ServiceTemplateRemove_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
```

```
        <data>
          <!-- ServiceTemplateID -->
          <value><i4>1000008</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                All selected Service Templates have been removed.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SetDefaultPaymentMethod_API

Description	Parameters	
	Passed	**Returned**
Method marks specified payment method as default for specified account. An exception returns if: • there is no specified account in PBA database; • there is no specified payment method in PBA database; • specified account is not an owner of the specified payment method.	2	1

Syntax

```
ErrorMessage BM::SetDefaultPaymentMethod_API(
  Int AccountID;
  Int PayToolID.
)
returns
  Str ErrorMessage -  message after payment method is marked as default:
"<PayToolID> will be used as default payment method."
```

Special Notes

- **Int** AccountID - ID of customer account, owner of the payment method. Must exist already in the PBA;
- **Int** PayToolID - ID of customer payment method to be marked as default.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SetDefaultPaymentMethod_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>1000002</i4></value>
         <!-- PayToolID -->
         <value><i4>7</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Credit Card ****1111 will be used as Default Payment Method.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SetResourceUsage_API

Description	Parameters	
	Passed	Returned
Method sets specified usage for the resources in the subscription. **Note**: before using this method, please make sure that the option "Allow Manual Resource Usage Update" is set to "Yes" in the Service Gate Settings of the Dummy Gate.	3 or more	1

Syntax

```
ErrorMessage DUMMYGATE::SetResourceUsage_API(
  Int SubscriptionID;
  Int ResourceID_1;
  Double FullAmount_1;
  ...
  Int ResourceID_N;
  Double FullAmount_N.
)
returns
  Str ErrorMessage - message after resource usage has been updated: "Resources on
subscription # have been updated.".
```

Special Notes

Int SubscriptionID - subscription ID;

Int ResourceID - resource ID;

Double FullAmount - new value for resource usage amount.

Note: the method is also available in BM container (for backwards compatibility) but starting from PBA 5.4 will be only available on DUMMYGATE.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
```

```
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DUMMYGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>SetResourceUsage_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000014</i4></value>
         <!-- ResourceID -->
         <value><i4>10023</i4></value>
         <!-- FullAmount -->
         <value><double>-1.0000</double></value>
         <!-- ResourceID -->
         <value><i4>10024</i4></value>
         <!-- FullAmount -->
         <value><double>10.0000</double></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Resources on subscription #1000014 have been updated.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SetResourceUsagePerInstance_API

Description	Parameters	
	Passed	Returned
Method is designed for work with multiple cloud instances (servers). The method sets specified resource usage by particular instances for selected subscription.	5 or more	1

Syntax

```
ErrorMessage DUMMYGATE::SetResourceUsagePerInstance_API(
   Int SubscriptionID;
   Int ResourceID_1;
   Double FullAmount_1;
   Str InstanceTag_1;
   Str InstanceName_1;
   ...
   Int ResourceID_N;
   Double FullAmount_N.
   Str InstanceTag_N;
   Str InstanceName_N;
)
returns
  Str ErrorMessage - message after resource usage has been updated: "Resources on
subscription # have been updated.".
```

Special Notes

Int SubscriptionID - subscription ID;

Int ResourceID - resource ID;

Str InstanceTag - internal ID of the server which is using specified subscription;

Str InstanceName - server name, which is displayed in notifications and reports;

Double FullAmount - new value for resource usage amount.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DUMMYGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>SetResourceUsagePerInstance_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000014</i4></value>
         <!-- ResourceID -->
         <value><i4>10023</i4></value>
             <!-- FullAmount -->
         <value><double>10.0000</double></value>
         <!-- InstanceTag -->
         <value><string>PPS001</string></value>
         <!-- InstanceName -->
         <value><string>Mailserver01</string></value>
             <!-- Next resource per instance assignment -->
         <!-- ResourceID -->
         <value><i4>10023</i4></value>
         <!-- FullAmount -->
         <value><double>20.0000</double></value>
         <!-- InstanceTag -->
         <value><string>PPS002</string></value>
         <!-- InstanceName -->
         <value><string>Mailserver02</string></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Resources on subscription #1000014 have been updated.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SingleDomainNameAvailability_API

	Parameters	
Description	**Passed**	**Returned**
Method checks single domain name availability.	5	11

Syntax

```
ListResult DOMAINGATE::SingleDomainNameAvailability_API(
  Int OperationType;
  Str FullDomainName;
  Int PlanID;
  Str TransferKey;
  Int SortNo.
  )
returns
  Str FullDomainName - domain name + domain zone the way customer has input it;
  Str TechDomainName - domain name + domain zone the way it has been registered;
  Str FullDomainKernel - domain name the way customer has input it;
  Str TechDomainKernel - domain name the way it has been registered;
  Str FullDomainZone - domain zone the way customer has input it;
  Str TechDomainZone - domain zone the way it has been registered;
  Str PlanID - domain service plan ID;
  Str VendorID - vendor ID (provider or reseller);
  Str SuggestionLevel - indicates what domain name is registered: 0 - equally what
customer input, 1 - duplicated name, 2 - suggested available domain zones, 3 -
suggested domain kernel;
  Str  DomainStatus - indicates domain name check result: 0 - undefined check
result, 1 - unavailable domain name, 2 - available domain name;
  Str Message - message from respective registrar plug-in.
```

Special Notes

- **Int** OperationType - type of the operation to perform with a domain: 10 - register new domain, 20 - renew domain, 90 - transfer existing domain;
- **Str** FullDomainName - domain name + domain zone the way customer has input it;
- **Int** PlanID - domain service plan ID;
- **Str** TransferKey - transfer key (if applicable);
- **Int** SortNo - any integer. The parameter is required, but not used.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>SingleDomainNameAvailability_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OperationType -->
         <value><i4>10</i4></value>
         <!-- FullDomainName -->
         <value>domain.com</value>
         <!-- PlanID -->
         <value><i4>1</i4></value>
         <!-- TrasferKey -->
         <value></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
```

```
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Full Domain Name -->
                 <value><string>domain.com</string></value>
                 <!-- Tech Domain Name -->
                 <value><string>domain.com</string></value>
                 <!-- Full Domain Kernel -->
                 <value><string>domain</string></value>
                 <!-- Tech Domain Kernel -->
                 <value><string>domain</string></value>
                 <!-- Full Domain Zone -->
                 <value><string>com</string></value>
                 <!-- Tech Domain Zone -->
                 <value><string>com</string></value>
                 <!-- Plan ID -->
                 <value><string>1</string></value>
                 <!-- Vendor ID -->
                 <value><string>1</string></value>
                 <!-- Suggestion level -->
                 <value><string>0</string></value>
                 <!-- Domain Status -->
                 <value><string>2</string></value>
                 <!-- Message -->
                 <value><string>
                  Domain not registered
                 </string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
 </methodResponse>
```

SimpleDomainNamesAvailability_API

	Parameters	
Description	**Passed**	**Returned**
Method validates domain names according internal PBA rules. For example, whether domain name contains special characters.	4 or more	11

Syntax

```
ListResult DOMAINGATE::SimpleDomainNamesAvailability_API(
  Int OperationType;
  Int CountName;
  Str DomainName_1
  ...
  Str DomainName_N;
  Int CountAPlan;
  Int ActivePlanID_1
  ...
  Int ActivePlanID_N;
  Int CountIPlan;
  Int InactivePlanID_1
  ...
  Int InactivePlanID_N;
  Int CountTransferKey;
  Str TransferKey_1;
  ...
  Str TransferKey_N.
  )
returns
  Str FullDomainName - domain name + domain zone the way customer has input it;
  Str TechDomainName - domain name + domain zone the way it has been registered;
  Str FullDomainKernel - domain name the way customer has input it;
  Str TechDomainKernel - domain name the way it has been registered;
  Str FullDomainZone - domain zone the way customer has input it;
  Str TechDomainZone - domain zone the way it has been registered;
  Str PlanID - domain service plan ID;
  Str VendorID - vendor ID (provider or reseller);
  Str SuggestionLevel - indicates what domain name is registered: 0 - equally what
customer input, 1 - duplicated name, 2 - suggested available domain zones, 3 -
suggested domain kernel;
  Str  DomainStatus - indicates domain name check result: 0 - undefined check
result, 1 - unavailable domain name, 2 - available domain name;
  Str Message - message from respective registrar plug-in.
```

Special Notes

- **Int** OperationType - type of the operation to perform with a domain: 10 - register new domain, 20 - renew domain, 90 - transfer existing domain;
- **Int** CountName - number of domain names to be passed;
- **Str** DomainName - domain name with or without TLD;
- **Int** CountAPlan - number of service plans ID to be passed;
- **Int** ActivePlanID - ID of service plan for TLD customer selected to search domain name available in;
- **Int** CountIPlan - number of service plans ID additionally to be passed;
- **Int** InactivePlanID - ID of service plan for TLD to search available domain name in additionally;
- **Int** CountTransferKey - number of transfer keys to be passed;
- **Str** TransferKey - transfer key (if applicable).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>DOMAINGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>SimpleDomainNamesAvailability_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- OperationType -->
         <value><i4>10</i4></value>
         <!-- CountName -->
         <value><i4>1</i4></value>
        <!-- DomainName  1 -->
         <value>domain</value>
         <!-- CountAPlan -->
         <value><i4>1</i4></value>
         <!-- ActivePlanID -->
         <value><i4>1</i4></value>
         <!-- CountIPlan -->
         <value><i4>1</i4></value>
         <!-- InactivePlanID -->
         <value><i4>3</i4></value>
         <!-- CountTransferKey -->
         <value><i4>0</i4></value>
        <!-- TransferKey -->
         <value></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Full Domain Name -->
                 <value><string>domain.com</string></value>
                 <!-- Tech Domain Name -->
                 <value><string>domain.com</string></value>
                 <!-- Full Domain Kernel -->
                 <value><string>domain</string></value>
                 <!-- Tech Domain Kernel -->
                 <value><string>domain</string></value>
                 <!-- Full Domain Zone -->
                 <value><string>com</string></value>
                 <!-- Tech Domain Zone -->
                 <value><string>com</string></value>
                 <!-- Plan ID -->
                 <value><string>1</string></value>
                 <!-- Vendor ID -->
                 <value><string>1</string></value>
                 <!-- Suggestion level -->
                 <value><string>0</string></value>
                 <!-- Domain Status -->
                 <value><string>2</string></value>
                 <!-- Message -->
                 <value><string>
                  Domain not registered
                 </string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

StartAllSubscriptionsForAccount_API

	Parameters	
Description	**Passed**	**Returned**
The method starts all subscriptions of the specified account that are in the *Stopped* status.	3	1

Syntax

```
ListResult BM :: StartAllSubscriptionsForAccount_API (
 Int AccountID,
 Int ReasonID,
 Str ReasonDescription.
)
returns
 Str ErrorMessage.
```

Parameters Description

Input Parameters:

- **Int** AccountID – Identifier of the customer account the subscriptions of which are to be processed.
- **Int** ReasonID – Identifier of the system reason code provided as the reason for subscription start.
- **Str** ReasonDescription – Optional comment explaining the reason for the subscriptions start.

Output Parameter:

Str ErrorMessage – Message issued after the subscriptions of the specified account have been started: "Operation done."

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>StartAllSubscriptionsForAccount_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
             <!-- AccountID -->
             <value><i4>1000003</i4></value>
                    <!-- ReasonID -->
             <value><i4>3</i4></value>
                    <!-- ReasonDescription -->
             <value><string>Processed due to valid business
reasons</string></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done.</string></value>
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

StopAllSubscriptionsForAccount_API

Description	Parameters	
	Passed	**Returned**
This method stops all subscriptions of the specified account.	3	1

Syntax

```
ListResult BM :: StopAllSubscriptionsForAccount_API (
 Int AccountID,
 Int ReasonID,
 Str ReasonDescription.
)
returns
 Str ErrorMessage.
```

Parameters Description

Input Parameters:

- **Int** AccountID – Identifier of the customer account the subscriptions of which are to be processed.
- **Int** ReasonID – Identifier of the system reason code provided as the reason for the subscription stop.
- **Str** ReasonDescription – Optional comment explaining the reason for the subscriptions stopping.

Output Parameter:

Str ErrorMessage – Message after the subscriptions of the specified account have been stopped: "Operation done."

Example

Request:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>StopAllSubscriptionsForAccount_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
             <!-- AccountID -->
             <value><i4>1000003</i4></value>
                    <!-- ReasonID -->
             <value><i4>9</i4></value>
                    <!-- ReasonDescription -->
             <value><string>Processed due to valid business
reasons</string></value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done.</string></value>
            </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

StateBookByCountry_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of states/provinces for specified country.	2	2

Syntax

```
ListResult BM::StateBookByCountry_API(
  Str CountryID;
  Int SortNo.
)
returns
  Str State - two-letter state code;
  Str Name - state name;
```

Special Notes

- **Int** CountryID - two-letter country code;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (State) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>StateBookByCountry_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- CountryID -->
         <value>au</value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
```

```
               <array>
                <data>

                 <!-- State Code -->
                 <value><string>ACT</string></value>

                 <!-- State Name -->
                 <value><string>Australian Capital Territory</string></value>
                </data>
               </array>
               </value>
               <value>
                <array>
                 <data>

                  <!-- State Code -->
                  <value><string>NSW</string></value>

                  <!-- State Name -->
                  <value><string>New South Wales</string></value>
                </data>
               </array>
               </value>
      ...
               <value>
                <array>
                 <data>

                  <!-- State Code -->
                  <value><string>WA</string></value>

                  <!-- State Name -->
                  <value><string>Western Australia</string></value>
                 </data>
                </array>
               </value>
              </data>
             </array>
            </value>
           </data>
          </array>
         </value>
        </member>
       </struct>
      </value>
     </param>
    </params>
   </methodResponse>
```

StoreApplicationToPlanIDMapListGet_API

Description	Parameters	
	Passed	**Returned**

Method returns the list of relations between APS application and service plan in PBA. This plan will be added to shopping cart once user gets to online store through marketplace redirect that includes the APS application identifier.	2	3 or more

Syntax

```
ListResult STORES::StoreApplicationToPlanIDMapListGet_API(
  Int StoreID;
  Int SortNo.
)
returns
  Int StoreID - requested online store ID;
  Int ApplicationID - ID of APS application;
  Int PlanID - ID of service plan to be added to cart.
```

Special Notes

- **Int** StoreID - ID of the existing online store. The list of online stores can be accessed from the **Product Director** > **Online Store Manager** > **Stores** submenu of Navigation tree;;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>StoreApplicationToPlanIDMapListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- StoreID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
```

```
               <array>
                <data>

                 <!-- Store ID -->
                 <value><i4>1</i4></value>

                 <!-- Application ID -->
                 <value><string>http://wordpress.org/</string></value>

                 <!-- Sevice Plan ID -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data><
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

StoreOperationToPathMapListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of relations between operations defined in marketplace protocol and online store usage scenario.	2	3 or more

Syntax

```
ListResult STORES::StoreOperationToPathMapListGet_API(
  Int StoreID;
  Int SortNo.
)
returns
  Str OperationID - ID of marketplace operation:
CREATE_DOMAIN - register domain;
CREATE_CERT - buy SSL Certificate;
PURCHASE_APPLICATION_LICENSE - buy application license;
  Int StoreID - requested online store ID;
  Str PathID - ID of APS application;
```

Special Notes

- **Int** StoreID - ID of the existing online store. The list of online stores can be accessed from the **Product Director** > **Online Store Manager** > **Stores** submenu of Navigation tree;;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>StoreOperationToPathMapListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- StoreID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Operation ID -->
                 <value><string>CREATE_DOMAIN </string></value>
                 <!-- Store ID -->
                 <value><i4>1</i4></value>
                 <!-- Path ID -->
                 <value><string>DOMAINS_PATH</string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data><
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

StoreCategoryParentPlansAvailableListGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of plans specified sales category assigned to as up-sale category.	2	6

Syntax

```
ListResult BM::StoreCategoryParentPlansAvailableListGet_API(
  Int CategoryID;
  Int SortNo.
)
returns
  Int PlanID - parent service plan ID;
  Str(60) Name - parent service plan name;
  Str(1024) ShortDescription - parent service plan short description;
  Str(1024) LongDescription - parent service plan long description;
  Int SalesCategoryID - ID of sales category up-sale service plan assigned to;
  Int ChildLimit - maximum number of up-sale subscriptions set for up-sale
category (Maximum Up-sale Subscriptions parameter).
```

Special Notes

- **Int** CategoryID - sales category ID;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Plan ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>StoreCategoryParentPlansAvailableListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- CategoryID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>2</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>

                 <!-- Parent plan ID -->
                 <value><i4>3</i4></value>

                 <!-- Parent plan name -->
                 <value><string>Linux Basic</string></value>

                 <!-- Parent plan short description -->
                 <value><string>Get Hosting!!!</string></value>

                 <!-- Parent plan long description -->
                 <value><string>Get Hosting!!!</string></value>

                 <!-- Sales category ID -->
                 <value><i4>1</i4></value>

                 <!-- Child subscriptions limit -->
                 <value><i4>6</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

StoreLayoutsListGet_API

Description	Parameters	
	Passed	Returned
Method returns the list of layout templates for specified online store listed under the **Product Director** > **Online Store Manager** > **Layout Templates** submenu of Navigation tree.	2	2

Syntax

```
ListResult STORES::StoreLayoutsListGet_API(
  Int StoreID;
  Int SortNo.
)
returns
  Str LayoutID - ID of layout template;
  Str Template - layout template body.
```

Special Notes

- **Int** StoreID - ID of the existing online store. The list of online stores can be accessed from the **Product Director** > **Online Store Manager** > **Stores** submenu of Navigation tree;
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Layout ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>StoreLayoutsListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- StoreID -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
```

```
               <array>
                <data>
                 <!-- Layout ID -->
                 <value><string>about</string></value>
                 <!-- Template body -->
                 <value><string>
&lt;h2&gt;About CrucialServer.com&lt;/h2&gt;
&lt;br /&gt;Since 1996, CrucialServer.com has served more
than 300,001 customers, consistently ranking the Top domain
name provider and Web hosting company according to independent
industry sources. In addition to Hosting and Domains,
CrucialServer.com provides a full suite of Internet services,
including Email Hosting, DNS Management and Web Site Promotion.
CrucialServer.com provides services to the consumer as well as
a highly regarded team of Hosting Resellers. A complete snapshot
of our service offering can be found on our home page.&lt;br /&gt;
&lt;br /&gt;&lt;b&gt;Recognition&lt;/b&gt;&lt;br /&gt;&lt;br /&gt;
CrucialServer.com is regularly honored by the web hosting industry,
the media, and most importantly, by our customers. We are proud to
have been awarded #1 Web Host in the last 10 years.&lt;br /&gt;
&lt;br /&gt; CrucialServer.com&lt;br&gt; 1 Crucial Server
Rd&lt;br&gt;CrucialTown, DC 95382&lt;br /&gt;&lt;br /&gt;
                 </string></value>
                </data>
               </array>
              </value>
   ...
              <value>
               <array>
                <data>
                 <!-- Layout ID -->
                 <value><string>contact</string></value>
                 <!-- Template body -->
                 <value><string>
&lt;h2&gt;Contacts&lt;/h2&gt;&lt;br /&gt;
&lt;b&gt;Support 24 Hours a Day, 7 Days a Week&lt;/b&gt;&lt;br&gt;
Toll Free: 888-777-4321&lt;br&gt;
Local: 798-207-5152&lt;br /&gt;&lt;br /&gt;
&lt;b&gt;Sales &amp; Billing&lt;/b&gt;&lt;br&gt;
Toll Free: 877-777-6003&lt;br&gt;
Local: 201-792-4847&lt;br /&gt;&lt;br /&gt;
&lt;b&gt;Address&lt;/b&gt;
CrucialServer.com&lt;br&gt;
1 Crucial Server Rd&lt;br&gt;
CrucialTown, DC 95382:xc&lt;br /&gt;&lt;br /&gt;
                 </string></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data><
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

StorePackageListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of all available store packages listed under the **Product Director** > **Online Store Manager** > **Stores** submenu of Navigation tree.	1	7

Syntax

```
ListResult STORES::StorePackageListGet_API(
  Int SortNo.
)
returns
  Int StoreID - ID of store package;
  Int AccountID - ID of account, store package belongs to;
  Str Name - store package name;
  Str Description - store package description;
  Int IsCurrent - signifies whether store package is selected for editing: "0" -
store package is not selected for editing, "1" - store package is selected for
editing, made changes are applied after synchronization;
  Int IsActive - signifies whether store package is active: "0" - store package is
not available for sales, "1" - store package is available for sales;
  Int IsDefault - signifies whether store package is provider's default online
store. Default online store is displayed at address specified in the Synchronization
Settings menu: "0" -store package is not default, "1" - store package is default.
```

Special Notes

- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Store ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>StorePackageListGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SortNo -->
         <value><i4>0</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <value>
               <array>
                <data>
                 <!-- Store ID -->
                 <value><i4>1</i4></value>

                 <!-- Account ID -->
                 <value><i4>1</i4></value>

                 <!-- Store name -->
                 <value><string>Provider Inc. Store</string></value>

                 <!-- Store description -->
                 <value><string>Provider Inc. Store</string></value>

                 <!-- IsCurrent -->
                 <value><i4>1</i4></value>

                 <!-- IsActive -->
                 <value><i4>1</i4></value>

                 <!-- IsDefault -->
                 <value><i4>1</i4></value>
                </data>
               </array>
              </value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionAdd_API

Description	Parameters	
	Passed	Returned
Method creates new subscription based on given parameters. Subscription information is added to PBA database.	20	1

Syntax

```
ItemResult BM::SubscriptionAdd_API(
  Int ParentID;
  Str(40) SubscriptionName;
  Int Status;
  Int ServStatus;
  Int StartDate;
  Int ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int Period;
  Int PeriodType;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate.
)
returns
  Int SubscriptionID - created subscription ID.
```

Special Notes

- **Int** ParentID - ID of the parent subscription, if there is no parent subscription, pass NULL (-2147483648);
- **Str(40)** SubscriptionName - new subscription name;
- **Int** Status - parent subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40 - "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 - "Administrative Hold", 85 - "Credit Hold", 89 - "Credit and Administrative Hold");
- **Int** ServStatus - parent subscription service status (10 - ''Not Provisioned'', 20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Str** StartDate - the date of the parent subscription start in Unix date format;

- **Str** ExpirationDate - expiration date of the parent subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;
- **Int** PlanID - ID of subscription service plan;
- **Int** Period - subscription period duration (for example, 1 if subscription period is 1 year);
- **Int** PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 - Month(s), 3 - Year(s);
- **Double** SetupFee - subscription setup fee;
- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

Note: fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days. If the parameter is passed as undefined (-2), the refund period value is taken from a service plan specified in PlanID parameter;;
- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);
- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Parent  subscriptionID -->
         <value><i4>-2147483648</i4></value>
         <!-- SubscriptionName -->
         <value>mysubscription</value>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
        <!-- startDate -->
         <value><i4>1219708800</i4></value>
         <!-- ExpirationDate -->
         <value><i4>1251244800</i4></value>
         <!-- AccountID -->
         <value><i4>1000023</i4></value>
         <!-- PlanID -->
         <value><i4>123</i4></value>
         <!-- Period -->
         <value><i4>1</i4></value>
         <!-- PeriodType -->
         <value><i4>3</i4></value>
         <!-- SetupFee -->
         <value><double>10</double></value>
         <!-- SubscriptionFee -->
         <value><double>12</double></value>
         <!-- RenewalFee -->
         <value><double>8</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
         <!-- NonRefundableAmt -->
         <value><double>10</double></value>
         <!-- RefundPeriod -->
         <value><i4>0</i4></value>
         <!-- BillingPeriodType -->
         <value><i4>2</i4></value>
         <!-- BillingPeriod -->
         <value><i4>1</i4></value>
         <!-- LastBillDate -->
         <value><i4>1219708800</i4></value>
         <!-- NextBillDate -->
         <value><i4>1222387200</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
            <!-- Added subscription ID -->
             <value><i4>1000038</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

SubscriptionDetailsGet_API

Description	Parameters	
	Passed	Returned
Method returns details of specified subscription.	1	7

Syntax

```
ItemResult BM::SubscriptionDetailsGet_API(
  Int SubscriptionID.
)
returns
  Int SubscriptionID - subscription ID;
  Str(40) SubscriptionName - subscription name;
  Int AccountID - ID of account the subscription belongs to;
  Int PlanID - subscription service plan ID;
  Str(60) PlanName - subscription service plan name. Starting PBA 5.0 a service
plan name is a multi-language field. To get a plan name in particular language
you need to specify this language in the method call. If a language is not
specified, the method returns value in English (default) language. For details,
refer to the Specifying Locale (on page 25) section;
```

```
  Int Status - subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40
- "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 -
"Administrative Hold", 85 - "Credit Hold", 89 - "Credit and  Administrative
Hold");
  Int ServStatus - subscription service current status. Possible values: 10 - Not
Provisioned, 20 - Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 -
Stopping, 70 - Removing, 80 - Changing Plan, 90 - Removed.
```

Special Notes

Int SubscriptionID - subscription ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000009</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
```

```
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Subscription ID -->
              <value><i4>1000009</i4></value>
              <!-- Subscription name -->
              <value><string>ID1000009</string></value>
              <!-- Account ID -->
              <value><i4>1000002</i4></value>
              <!-- Plan ID -->
              <value><i4>3</i4></value>
              <!-- Plan name -->
              <value><string>Linux Basic</string></value>
              <!-- Subscription status -->
              <value><i4>30</i4></value>
              <!-- Subscription service status -->
              <value><i4>50</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionDetailsGetEx_API

Description	Parameters	
	Passed	Returned
Method returns extended set of details for a given subscription.	1	13

Syntax

```
ItemResult BM::SubscriptionDetailsGetEx_API(
  Int SubscriptionID.
)
returns
  Int SubscriptionID – subscription identifier;
  Str SubscriptionName - subscription name;
  Int AccountID - identifier of the account to which the subscription belongs;
  Int PlanID - identifier of the service plan with which the subscription is
associated;
  Str PlanName - name of the subscription service plan. Starting from PBA 5.0, a
service plan name is a multi-language field. To get a plan name in a particular
language, you need to specify this language in the method call. If a language is
not specified, the method returns values in English (default) language. For
details, refer to the Specifying Locale (on page 25) section;
  Int Status - subscription status (the possible status values are: 10 - "Ordered",
15 - "Trial", 30 - "Active", 40 - "Graced", 50 - "Expired", 60 - "Terminated", 70
- "Canceled", 80 - "Administrative Hold", 85 - "Credit Hold", 89 - "Credit and
Administrative Hold");
  Int ServStatus - subscription service current status (the possible status values
are: 10 - Not Provisioned, 20 - Provisioning, 30 - Stopped, 40 - Starting, 50 -
Running, 60 - Stopping, 70 - Removing, 80 - Changing Plan, 90 - Removed);
  Int StartDate - start date of the subscription period (in Unix format);
  Int ExpirationDate – expiration date of the subscription period (in Unix
format);
  Int LastBillDate – date of when the subscription was billed the last time (Unix
format);
  Int NextBillDate – date of when the subscription is to be billed the next time
(Unix format);
• Int BillingPeriodType - subscription billing period type (4 - monthly on
statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
• Int BillingPeriod - subscription billing period duration (if BillingPeriodType=2
and BillingPeriod=1, subscription billing period is 1 month).
```

Special Notes

Int SubscriptionID – ID of subscription of interest.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionDetailsGetEx_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000009</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Subscription ID -->
              <value><i4>1000009</i4></value>
              <!-- Subscription name -->
              <value><string>ID1000009</string></value>
              <!-- Account ID -->
              <value><i4>1000002</i4></value>
              <!-- Plan ID -->
              <value><i4>3</i4></value>
              <!-- Plan name -->
              <value><string>Linux Basic</string></value>
              <!-- Subscription status -->
              <value><i4>30</i4></value>
              <!-- Subscription service status -->
              <value><i4>50</i4></value>
                <!-- Subscription start date -->
              <value><i4>1337264240</i4></value>
                <!-- Subscription expiration date -->
                  <value><i4>1368800240</i4></value>
                <!-- Subscription last billing date -->
              <value><i4>1337264240</i4></value>
                <!-- Subscription next billing date -->
                  <value><i4>1368800240</i4></value>
              <!-- BillingPeriodType -->
              <value><i4>2</i4></value>
              <!-- BillingPeriod -->
              <value><i4>1</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionListByResourceRateGet_API

Description	Parameters	
	Passed	**Returned**
Method returns the list of subscriptions, with specified resource rate.	2	7

Syntax

```
ListResult BM::SubscriptionListByResourceRateGet_API(
  Int PlanRate;
  Int SortNo.
}
returns
  Int SubscriptionID - subscription ID;
  Str(40) SubscriptionName - subscription name;
  Int AccountID - ID of account the subscription belongs to;
  Int PlanID - subscription service plan ID;
  Str(60) Plan - subscription service plan name;
  Int Status - subscription current status. Possible values: 10 - Ordered, 15 -
Trial, 30 - Active, 40 - Graced, 50 - Expired, 60 - terminated, 70 - Canceled, 80
- Suspended;
  Int ServStatus - subscription service current status. Possible values: 10 - Not
Provisioned, 20 - Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 -
Stopping, 70 - Removing, 80 - Changing Plan, 90 - Removed.
```

Special Notes

- **Int** PlanRate - ID of service plan resource rate.
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 2 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Subscription ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionListByResourceRateGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- PlanRate -->
         <value><i4>1</i4></value>
         <!-- SortNo -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- Subscription ID -->
              <value><i4>1000009</i4></value>
              <!-- Subscription name -->
              <value><string>ID1000009</string></value>
              <!-- Account ID -->
              <value><i4>1000002</i4></value>
              <!-- Plan ID -->
              <value><i4>3</i4></value>
              <!-- Plan name -->
              <value><string>Linux Basic</string></value>
              <!-- Subscription status -->
              <value><i4>30</i4></value>
              <!-- Subscription service status -->
              <value><i4>50</i4></value>
             </data>
            </array>
           </value>
   ...
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionResourcesListGet_API

	Parameters	
Description	**Passed**	**Returned**
Method returns the list of resources included or available for specified subscription.	2	24

Syntax

```
ListResult BM::SubscriptionResourcesListGet_API(
  Int SubscriptionID;
  Int SortNo.
}
returns
  Int ResourceID - ID of a resource;
  Int ResourceRateID - ID of respective resource rate;
  Str ResourceName - name of a resource;
  Str StoreDescription - description of resource displayed in online store;
  Str StorePriceText - description of resource rate displayed in online store;
  Int StoreSortOrder - resource sort number in resources list displayed in online
store;
  Str Status - resource current status: Purchased - if subscription is not active,
the status means that resource is included. If subscription is active, the status
means that resource is purchased for subscription; Provisioning - the resource
will be installed soon; Installed - the resource is installed for subscription;
Failed - some technical problems occurred while order processing or resource
installation; Ordered - the resource is ordered for subscription; Not Included -
status is obsolete, similar to Optional; Removing - the resource is removing from
subscription; Disabled - the resource is disabled in case of subscription
cancellation; Optional - the resources can be purchased to subscription
additionally;
  Str Resource Category - the resource category a resource belongs to;
  Double IncludedAmount - resource amount included in subscription fee;
  Double AdditionalAmount - resource additional amount available;
  Double UsedAmount - resource used amount;
  Double OrderedAmount - ordered resource amount;
  Str Unit - resource unit of measure;
  Double MinUnits - minimal amount of resource;
  Double MaxUnits - maximal amount of resource;
  Int Measurable - signifies whether resource is measurable ("1") or not ("0");
  Str RelativeStatus - signifies how a resource was obtained: Standard - the
resource was added to the subscription upon service plan purchase; Switch Plan -
the resource was added to the subscription when subscription plan was changed;
Additional - the resource was added to the plan after the subscription had been
provisioned; Sync With Plan - the plan was changed after the subscription had
been provisioned and synchronized with the plan; Upgrade / Downgrade - additional
resource units were ordered after the initial resources provisioning;
  Str OrderNumber - the number of order placed for resource purchase, relevant
only for resources in Ordered status. This field is empty for resources in other
statuses;
  Double SetupFee - resource setup fee;
  Double RecurringFee - resource recurring fee;
  Double OveruseFee - resource overusage fee;
  Str Location - signifies whether resource in subscription is synchronized with
service plan: "Subscription" - resource exists only in subscription, in service
plan its resource rate was removed; "ServicePlan" - resource rate exists in
service plan for this resource, but resource is not purchased for subscription;
"Both" - resource exists both in service plan and subscription;
  Int IsOveruseFeeTiered - shows whether price tiers are configured for a overuse
fee of the corresponding object (resource rate, resource rate period, resource in
subscription), or not (1) or not (0);
  Int IsRecurringFeeTiered - shows whether price tiers are configured for a
recurring fee of the corresponding object (resource rate, resource rate period,
resource in subscription), or not (1) or not (0).
```

Special Notes

- **Int** SubscriptionID - ID of the subscription the resources list is requested.
- **Int** SortNo - defines how output data is sorted: 1 - the output is sorted by first column in ascending order, 1 - the output is sorted by second column in ascending order, etc. Negative value makes output sorted in descending order, for example: -1 - the output is sorted by first column (Resource ID) in descending order.

Example

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
 <methodCall>
  <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value><string>BM</string></value>
      </member>
      <member>
       <name>Method</name>
       <value><string>
        SubscriptionResourcesListGet_API
       </string></value>
      </member>
      <member>
       <name>Params</name>
        <value>
         <array>
          <data>
           <!-- Subscription ID -->
           <value><i4>1000001</i4></value>
           <!-- Sort No -->
           <value><i4>0</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value>
              <array>
               <data>
                <!-- Resource ID -->
                <value><i4>1</i4></value>
                <!-- Resource rate ID -->
                <value><i4>4</i4></value>
                <!-- Resource name -->
                <value><string>
                 Number of Subscriptions
                </string></value>
                <!-- Store description -->
                <value><string>
                 Number of subscriptions
                </string></value>
                <!-- Store price text -->
                <value><string>
                 5 subscriptions for free
                </string></value>
                <!-- Store sort order -->
                <value><i4>999</i4></value>
                <!-- Resource status -->
                <value><string>Installed</string></value>
                <!-- Resource category -->
                <value><string>
                 Additional resources
                </string></value>
                <!-- Resource included amount -->
                <value><double>1000.000000</double></value>
                <!-- Resource additional amount -->
                <value><double>0.000000</double></value>
                <!-- Resource used amount -->
                <value><double>0.000000</double></value>
                <!-- Resource ordered amount -->
                <value><double>0.000000</double></value>
                <!-- Resource UOM -->
                <value><string>Item</string></value>
                <!-- Resource min units -->
                <value><double>0.000000</double></value>
                <!-- Resource max units -->
                <value><double>1000.000000</double></value>
                <!--Measurable -->
                <value><i4>0</i4></value>
                <!-- Resource relative status -->
                <value><string>Standard</string></value>
                <!-- Order number -->
                <value><string> </string></value>
                <!-- Resource setup fee -->
                <value><double>0.000000</double></value>
                <!-- Resource recurring fee -->
                <value><double>0.000000</double></value>
                <!-- Resource overuse fee -->
                <value><double>0.000000</double></value>
                <!-- Resource location -->
                <value><string>Both</string></value>
                <!-- IsOveruseFeeTiered -->
                <value><i4>0</i4></value>
                <!-- IsRecurringFeeTiered -->
                <value><i4>0</i4></value>
               </data>
              </array>
              </value>
...
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

SubscriptionStart_API

Description	Parameters	
	Passed	**Returned**
Method starts specified subscription service. Service is started, if its current status is **Stopped**.	3	1

Syntax

```
ErrorMessage BM::SubscriptionStart_API(
  Int SubscriptionID;
  Int ReasonID;
  Str Descr.
)
returns
  Str ErrorMessage - message after subscription service has been started: "Service
of Subscription # has been started."
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** ReasonID - ID of system reason code to be stated as reason of subscription starting. Available reason codes can be accessed from **Configuration Director** > **Miscellaneous Settings** > **Reason Codes** submenu of the Navigation tree (**Reason Code** column);
- **Str** Descr - comment on the reason subscription service is started.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionStart_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000009</i4></value>
         <!-- ReasonID -->
         <value><i4>1</i4></value>
         <!-- Descr -->
         <value>Subscription is started due to customer request.</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
```

```
             <member>
              <name>Status</name>
               <value><string>
                Service of Subscription #1000009 has been started.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionStatusUpdate_API

Description	Parameters	
	Passed	**Returned**
Method sets selected subscription statuses to specified values. If you pass NULL for some status it is remained unchanged. To pass NULL, use '-2147483648' value.	3	1

Syntax

```
ErrorMessage BM::SubscriptionStatusUpdate_API(
  Int SubscriptionID;
  Int Status;
  Int ServStatus.
)
returns
  Str ErrorMessage - message after subscription statuses has been updated:
"Operation done."
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** Status - subscription status (10 - Ordered, 15 - Trial, 30 - Active, 40 - Graced, 50 - Expired, 60 - Terminated, 70 - Canceled, 80 - Suspended);
- **Int** ServStatus - subscription service status (10 - Not Provisioned, 20 - Provisioning, 30 - Stopped, 40 - Starting, 50 - Running, 60 - Stopping, 70 - Removing, 80 - Changing Plan, 90 - Removed).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionStatusUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000009</i4></value>
         <!-- Status -->
         <value><i4>40</i4></value>
         <!-- ServStatus -->
         <value><i4>30</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
```

```
             <member>
              <name>Status</name>
               <value><string>
                Operation done.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionStop_API

	Parameters	
Description	**Passed**	**Returned**
Method stops service of specified subscription.	2	1

Syntax

```
ErrorMessage BM::SubscriptionStop_API(
  Int SubscriptionID;
  Str(2000) Comment.
)
returns
  Str ErrorMessage - message after subscription service has been stopped: "Service
of Subscription # has been stopped."
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Str(2000)** Comment - reason why subscription service is stopped.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionStop_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1000009</i4></value>
         <!-- Comment -->
         <value>Upgrade</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Service of Subscription #1000009 has been stopped.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscriptionWithIDAdd_API

	Parameters	
Description	**Passed**	**Returned**
Method creates new subscription based on given parameters with specified ID. Subscription information is added to PBA database.	21	1

Syntax

```
ErrorMessage BM::SubscriptionWithIDAdd_API(
  Int SubscriptionID;
  Int ParentID;
  Str(40) SubscriptionName;
  Int Status;
  Int ServStatus;
  Int startDate;
  Int ExpirationDate;
  Int AccountID;
  Int PlanID;
  Int Period;
  Int PeriodType;
  Double SetupFee;
  Double SubscriptionFee;
  Double RenewalFee;
  Double TransferFee;
  Double NonRefundableAmt;
  Int RefundPeriod;
  Int BillingPeriodType;
  Int BillingPeriod;
  Int LastBillDate;
  Int NextBillDate.
)
returns
  Str ErrorMessage -  message after subscription has been added: "Subscription is
added."
```

Special Notes

- **Int** SubscriptionID - subscription ID;
- **Int** ParentID - ID of the parent subscription, if there is no parent subscription, pass NULL (-2147483648);
- **Str(40)** SubscriptionName - new subscription name;
- **Int** Status - parent subscription status (10 - "Ordered", 15 - "Trial", 30 - "Active", 40 - "Graced", 50 - "Expired", 60 - "Terminated", 70 - "Canceled", 80 - "Administrative Hold", 85 - "Credit Hold", 89 - "Credit and Administrative Hold");

- **Int** ServStatus - parent subscription service status (20 - ''Provisioning'', 30 - ''Stopped'', 40 - ''Starting'', 50 - ''Running'', 60 - ''Stopping'', 70 - ''Removing'', 80 - ''Changing Plan'', 90 - ''Removed'');
- **Str** StartDate - the date of the parent subscription start in Unix date format;
- **Str** ExpirationDate - expiration date of the parent subscription in Unix date format. *Expiration Date = Start Date + Subscription Period*;
- **Int** AccountID - ID of the customer account;
- **Int** PlanID - ID of subscription service plan;
- **Int** Period - subscription period duration (for example, 1 if subscription period is 1 year);
- **Int** PeriodType - subscription period type: 0 - Hour(s), 1 - Day(s), 2 - Month(s), 3 - Year(s);
- **Double** SetupFee - subscription setup fee;
- **Double** SubscriptionFee - subscription recurring fee;
- **Double** RenewalFee - subscription renewal fee;
- **Double** TransferFee - subscription transfer fee;
- **Double** NonRefundableAmt subscription non refundable sum;

Note: fees for subscription are passed in "00.00" format and provider's default currency.

- **Int** RefundPeriod - subscription refund period duration in days. If the parameter is passed as undefined (-2), the refund period value is taken from a service plan specified in PlanID parameter;
- **Int** BillingPeriodType - subscription billing period type (4 - monthly on statement cycle date, 2 - fixed number of months, 3 - fixed number of years);
- **Int** BillingPeriod - subscription billing period duration (if BillingPeriodType=2 and BillingPeriod=1, subscription billing period is 1 month);
- **Int** LastBillDate - subscription last billing date in Unix date format;
- **Int** NextBillDate - subscription next billing date in Unix date format.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscriptionWithIDAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1023</i4></value>
         <!-- Parent  subscriptionID -->
         <value><i4>-2147483648</i4></value>
         <!-- SubscriptionName -->
         <value>mysubscription</value>
         <!-- Status -->
         <value><i4>30</i4></value>
         <!-- ServStatus -->
         <value><i4>50</i4></value>
         <!-- startDate -->
         <value><i4>1219708800</i4></value>
         <!-- ExpirationDate -->
         <value><i4>1251244800</i4></value>
         <!-- AccountID -->
         <value><i4>10000011</i4></value>
         <!-- PlanID -->
         <value><i4>12</i4></value>
         <!-- Period -->
         <value><i4>1</i4></value>
         <!-- PeriodType -->
         <value><i4>3</i4></value>
         <!-- SetupFee -->
         <value><double>10</double></value>
         <!-- SubscriptionFee -->
         <value><double>12</double></value>
         <!-- RenewalFee -->
         <value><double>5</double></value>
         <!-- TransferFee -->
         <value><double>0</double></value>
         <!-- NonRefundableAmt -->
         <value><double>10</double></value>
         <!-- RefundPeriod -->
         <value><i4>0</i4></value>
         <!-- BillingPeriodType -->
         <value><i4>2</i4></value>
         <!-- BillingPeriod -->
         <value><i4>1</i4></value>
         <!-- LastBillDate -->
         <value><i4>1219708800</i4></value>
         <!-- NextBillDate -->
         <value><i4>1222387200</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Subscription is added.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscrParamValueGet_API

Description	Parameters	
	Passed	**Returned**
The method returns a value of the required parameter for a specified subscription.	2	1

Syntax

```
ErrorMessage BM::SubscrParamValueGet_API(
  Int SubscriptionID;
  Str ParameterID;
  )
returns
  Str ParameterValue.
```

Parameters Description

Input Parameters:

- **Int** SubscriptionID – Identifier of an existing subscription.
- **Str** ParameterID – The subscription parameter the current value of which you want to be returned (for example, a *DomainID*).

Output Parameter:

Str ParameterValue – Value of the requested subscription parameter.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscrParamValueGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
             <!-- SubscriptionID -->
         <value><i4>1030465</i4></value>
             <!-- ParameterID -->
         <value><string>DomainID</string></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
                     <!-- ParamaterValue -->
                <value><string>domain.example.com</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SubscrParamValueUpdate_API

Description	Parameters	
	Passed	**Returned**
Method updates specified subscription parameters with new values.	3	1

Syntax

```
ErrorMessage BM::SubscrParamValueUpdate_API(
  Int SubscriptionID;
  Str ParameterID;
  Str ParameterValue;
)
returns
  Str ErrorMessage - message after specified subscription parameters have been
updated: "Operation done."
```

Special Notes

- **Int** SubscriptionID - ID of existing subscription, which parameters are to be updated;
- **Str** ParameterID - subscription parameter to be set (e.g., DomainID);
- **Str** ParameterValue - the new value of selected parameter.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>SubscrParamValueUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- SubscriptionID -->
         <value><i4>1030465</i4></value>
         <!-- ParameterID -->
         <value><string>DomainID</string></value>
         <!-- ParamaterValue -->
         <value><string>domain.example.com</string></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

SyncDomainIDParameterToPBA

	Parameters	
Description	**Passed**	**Returned**
Method takes the first domain name assigned to the POA subscription and saves its name as a *DomainID* parameter into PBA subscription.	0	1

Syntax

```
ErrorMessage PEMGATE::SyncDomainIDParameterToPBA(void)
returns
  Str ErrorMessage - message on assigning PBA subscription names: "Operation
done.".
```

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>PEMGATE</value>
     </member>
     <member>
      <name>Method</name>
      <value>SyncDomainIDParameterToPBA</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
           </struct>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

TokenForAccountGet_API

Description	Parameters	
	Passed	**Returned**
This method creates a one-time authorization *token for* a specified customer. This token can be used further as the *Token=<token_returned_by_PBA>* HTTP post parameter along with the *Type=login* operation when redirecting a customer to the PBA-E Online Store.	1	1

Syntax

```
BM::TokenForAccountGet_API(
  Int AccountID.
)
returns
  Str Token.
```

Parameters Description

Input parameter:

Int AccountID – PBA identifier of the customer for whom you want to get a token.

Output parameter:

Str Token – One-time authorization token created for the specified customer.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
does not parse XML comments, so REMOVE all the comments from an actual request. -->
<methodCall>
 <methodName>Execute</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Server</name>
       <value>BM</value>
      </member>
      <member>
       <name>Method</name>
       <value>TokenForAccountGet_API</value>
      </member>
      <member>
       <name>Params</name>
       <value>
        <array>
         <data>
          <!--AccountID --!>
          <value><i4>ACCOUNT_ID</i4></value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
```

```
            <value>
            <!--Token --!>
<string>zHaQ2mfiNEtMcWuMLEuEcZCOq5BgqR2x04Fk7YJzjIpj6drwJszpZOiSBvexAMbtG5nJiMQ6T
jD0UBOp</string>
            </value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

UniqueGroupDetailsGet_API

Description	Parameters	
	Passed	Returned
Method returns details of specified unique group.	1	3

Syntax

```
ItemResult BM::UniqueGroupDetailsGet_API(
  Int GroupID.
)
returns
  Int GroupID - ID of the unique group;
  Str(30) Name - unique group name;
  Str(1024) Description - description of the unique group.
```

Special Notes

Int GroupID - ID of the unique group.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UniqueGroupDetailsGet_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- GroupID -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <!-- GroupID -->
              <value><i4>1</i4></value>

              <!-- Group Name -->
              <value><string>Domain Registration</string></value>

              <!-- Group Description -->
              <value><string>Groups domain service plans.</string></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UniversalPaymentMethodAdd_API

Description	Parameters	
	Passed	**Returned**
This method is used to add a universal payment method to a specified customer account (to assign to it a corresponding payment system).	The number of passed parameters depends on the number of created specific attributes that the payment method has.	1

Syntax

```
ItemResult BM::UniversalPaymentMethodAdd_API(
  Int AccountID;
  Str PaySystem;
  Int UseForAutoPayments;
  Str Param1_ID;
  Str Param1_Value;
  ...
  Str ParamN_ID;
  Str ParamN_Value;
)
returns
  Int AC_CODE.
```

Parameters Description

Input parameters:

- **Int** AccountID - Identifier to which you want to assign a universal payment method.
- **Str** PaySystem - Identifier of the payment system that is to be assigned to the specified account.
- **Int** UseForAutoPayments - This parameter defines whether to use the created payment method to process auto-payments ot not. The parameter values are as follows:
  - 1 - the UseForAutoPayments option is set to *Yes*.
  - 0 - the UseForAutoPayments option is set to *No*.
- **Str** Param1_ID - Identifier of attribute 1.
- **Str** Param1_Value - Value of attribute 1.
- ...
- **Str** ParamN_ID - Identifier of attribute N.
- **Str** ParamN_Value - Value of attribute N.

Output parameter:

Int AC_CODE - Identifier of the payment method (which contains information about the customer account and the payment system associated to it) in PBA.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the xml comments. REMOVE all the comments from an actual request -->
    <methodCall>
      <methodName>Execute</methodName>
        <params>
          <param>
            <value>
              <struct>
                <member>
                  <name>Server</name>
                  <value>BM</value>
                </member>
                <member>
                   <name>Method</name>
                   <value>UniversalPaymentMethodAdd_API</value>
                </member>
                <member>
                   <name>Params</name>
                   <value>
                    <array>
                      <data>
                        <!-- Identifier of the account to which you want to assign the payment method. -->
                        <value><i4>1000001</i4></value>
                        <!-- Identifier of the Payment System.-->
                        <value><string>WiredPhoneAcct</string></value>
                        <!-- A parameter defining whether to use the method for AutoPayments or not.-->
                        <value><i4>0</i4></value>
                        <!-- Param1 ID-->
                        <value><string>PhoneNbr</string></value>
                        <!-- Param1 Value-->
                        <value><string>1.123.123456789</string></value>
                        <!-- ParamN ID-->
                        <value><string>-11ddr</string></value>
                        <!-- ParamN Value-->
                        <value><string>+100.003.1234567</string></value>
                      </data>
                    </array>
                   </value>
                </member>
              </struct>
            </value>
          </param>
       </params>
    </methodCall>
```

Response

```
<?xml version="1.0"?>
```

```
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
            <value><i4>11</i4></value>
           </data>
          </array>
         </value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodResponse>
```

UpdateObjAttrList_API

Description	Parameters	
	Passed	Returned
The method adds new attributes and updates the values of the already assigned ones for the specified account or user.	4 or more	1

Syntax

```
ErrorMessage BM::UpdateObjAttrList_API(
  Int AttributeType;
  Int ObjID;
  Str AttributeID_1;
  Str AttributeID_1_Value;
  ...
  Str AttributeID_N;
  Str AttributeID_N_Value.
)
returns
  Str ErrorMessage.
```

Parameters Description

Input Parameters:

- **Int** AttributeType – Parameter that defines the attribute type. The parameter values and their meanings are as follows:
  - 0 – available for accounts.
  - 1 – available for users.
- **Int** ObjID – Identifier of an account (a provider, customer or reseller) or a user identifier.

  Important: If you set *AttributeType* to 0 (available for accounts), specify the account ID, not the user ID, as a value of the *ObjID* parameter, and vice versa.

- **Str** AttributeID – Identifier of the existing attribute.
- **Str** Value – New value of the attribute.

Output Parameter:

Str ErrorMessage – Message issued after the list of attributes for the account or user has been updated. The message content is the following: "Operation done."

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are given only for your convenience. PBA XMLRPC
 does not parse the XML comments, REMOVE all the comments from an actual request.-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UpdateObjAttrList_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AttributeType -->
         <value><i4>0</i4></value>
         <!-- ObjID -->
         <value><i4>1000002</i4></value>
         <!-- AttributeID -->
         <value>ICQ</value>
         <!-- Value -->
         <value>235568985</value>
         <!-- AttributeID -->
         <value>DOB</value>
         <!-- Value -->
         <value>16.02.1984</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
```

```
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UpdateStore_API

Description	Parameters	
	Passed	Returned
Method updates specified online store parameters with new values.	9	1

Syntax

```
ErrorMessage STORES::UpdateStore_API(
  Int StoreID;
  Int LogoBlobDocID;
  Str ColorPalette;
  Str Tagline;
  Str AboutUs;
  Str WelcomText;
  Str Keywords;
  Str Description;
  Str CurrencyID;
)
returns
  Str ErrorMessage - message after specified online store parameters have been
updated: "Operation done."
```

Special Notes

- **Int** StoreID - ID of existing online store;
- **Int** LogoBlobDocID - ID of logo image in Blobdocument table;
- **Str** ColorPalette - store color palette name, default values are "general" and "crucial". Online store general parameter COLOR_PALETTE;
- **Str** Tagline - store slogan. Online store general parameter TAGLINE;
- **Str** AboutUs - content of store standard layout template `about.tpl`;
- **Str** WelcomText - introductory text on the home page. Online store general parameter WELCOM_TEXT;
- **Str** Keywords - content of meta tag "keywords" of store page header, used by search engines. Online store general parameter KEYWORDS;
- **Str** Description - online store description. Online store general parameter DESCRIPTION;
- **Str** CurrencyID - three-letter currency code to be used in online store. Online store general parameter CURRENCY_ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>STORES</value>
     </member>
     <member>
      <name>Method</name>
      <value>UpdateStore_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- StoreID -->
         <value><i4>5</i4></value>
         <!-- LogoBlobDocID -->
         <value><i4>10007</i4></value>
         <!-- ColorPalette -->
         <value>crucial</value>
         <!-- Tagline -->
         <value>Best Web Hosting</value>
         <!-- AboutUs -->
         <value>&lt;h2&gt;About MyStore.com&lt;/h2&gt;</value>
         <!-- WelcomText -->
         <value>Welcome to my web store</value>
         <!-- Keywords -->
         <value>hosting billing domain free</value>
         <!-- Description -->
         <value>my store description</value>
         <!-- CurrencyID -->
         <value>USD</value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
```

```
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserAdd_API

Description	Parameters	
	Passed	**Returned**
Method creates new user in the system. If login and password are empty, the user can be used only as contact information.	23	1

Syntax

```
ItemResult BM::UserAdd_API(
  Int AccountID;
  Int ExternalID;
  Str(64) Login;
  Str(32) Password;
  Str(30) FName;
  Str(30) MName;
  Str(30) LName;
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention;
  Int AddFARole.
)
returns
  Int UsersID - created user ID.
```

Special Notes

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the Password argument must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42). Avoiding sensitive data is obligatory, otherwise request will fail.

- **Int** AccountID - customer account ID;
- **Int** ExternalID - user's identifier for external system. If there is no ID for external system, should be passed '0';
- **Str(64)** Login - customer user login;

- **Str(32)** Password - user's password;
- **Str(30)** FName - user contact information: first name;
- **Str(30)** MName - user contact information: middle name;
- **Str(30)** LName - user contact information: last name;
- **Str(100)** Email - user contact information: e-mail address;
- **Str(80)** Address1 - user contact information: address (line 1/2);
- **Str(80)** Address2 - user contact information: address (line 2/2);
- **Str(40)** City - user contact information: city;
- **Str(80)** State - user contact information: state or province;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(4)** PhCountryCode - user contact information: phone country code;
- **Str(10)** PhAreaCode - user contact information: phone area code;
- **Str(20)** PhNumber - user contact information: phone number;
- **Str(10)** PhExtention - user contact information: phone extension;
- **Str(4)** FaxCountryCode - user contact information: fax country code;
- **Str(10)** FaxAreaCode - user contact information: fax area code;
- **Str(20)** FaxNumber - user contact information: fax number;
- **Str(10)** FaxExtention - user contact information: fax extension.
- **Int** AddFARole - signifies whether Full Access role is assigned to created user (0 - No, 1 - Yes).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>100023</i4></value>
         <!-- ExternalID -->
         <value><i4>1</i4></value>
         <!-- Login -->
         <value>admin</value>
         <!-- Password -->
         <value>XXXsetup</value>
         <!-- FName -->
         <value>John</value>
         <!-- MName -->
         <value>F</value>
         <!-- LName -->
         <value>Smith</value>
         <!-- Email -->
         <value>test@mail.ru</value>
         <!-- Address1 -->
         <value>Green Valley</value>
         <!-- Address2 -->
         <value>223</value>
         <!-- City -->
         <value>New York</value>
         <!-- State -->
         <value>NY</value>
         <!-- Zip -->
         <value>12345</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PhCountryCode -->
         <value>1</value>
         <!-- PhAreaCode -->
         <value>1234</value>

         <!-- PhNumber -->
         <value>123</value>

         <!-- PhExtention -->
         <value>123</value>

         <!-- FaxCountryCode -->
         <value>1</value>

         <!-- FaxAreaCode -->
         <value>1234</value>

         <!-- FaxNumber -->
         <value>1234</value>

         <!-- FaxExtention -->
         <value>124</value>

        <!-- AddFARole -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
            <data>
             <!-- Added Account ID -->
             <value><i4>1000002</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

UserDetailsGet_API

	Parameters	
Description	**Passed**	**Returned**
Method performs a search of a user details by the entered identifier of this user.	22	1

Syntax

```
BM::UserDetailsGet_API(
  Int UserID;
)
returns
  Int UserID;
  Str(40) Login;
  Int AccountID;
  Str(30) FName;
  Str(30) MName;
  Str(30) LName;
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention;
  Int Status.
```

Parameters description

Input parameter:

- **Int** UserID - Identifier of the user about which you want to get information.

Output parameters:

- **Int** UserID - Identifier of the specified user.
- **Str(40)** Login - Login name of a user.
- **Int** AccountID - Identifier of the specified account.
- **Str(30)** FName - First name of the user.
- **Str(30)** MName - Middle name of the user.
- **Str(30)** LName - Last name of the user.
- **Str(100)** Email - Email address of the user.
- **Str(80)** Address1 - Primary address of the user.
- **Str(80)** Address2 - Additional address of the user.
- **Str(40)** City - Residential city of the user.
- **Str(80)** State - Residential state of the user.
- **Str(10)** Zip - Zip code of the user residence.
- **Str(2)** CountryID - Code of the residential country of the user.
- **Str(4)** PhCountryCode - Phone code of the user residential country.
- **Str(10)** PhAreaCode - Phone code of the user residential area.
- **Str(20)** PhNumber - User phone number.
- **Str(10)** PhExtention - User phone number extension.
- **Str(4)** FaxCountryCode - Fax code of the user residential country.
- **Str(10)** FaxAreaCode - Fax code of the user residential area.
- **Str(20)** FaxNumber - User fax number.
- **Str(10)** FaxExtention - Extension of the user fax number.

- **Int** Status - Status of the user account. The status can be:
  - **0** if the user is enabled.
  - **1** if the user is disabled.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
 <methodCall>
  <methodName>Execute</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>Server</name>
        <value>BM</value>
       </member>
       <member>
        <name>Method</name>
        <value>UserDetailsGet_API</value>
       </member>
       <member>
        <name>Params</name>
        <value>
         <array>
          <data>
           <!-- UserID -->
           <value><i4>1000001</i4></value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
```

Response

```
<?xml version="1.0"?>
 <methodResponse><params><param>
  <value>
   <struct>
    <member>
     <name>Result</name>
      <value>
       <array>
        <data>
         <value>
          <array>
           <data>
           <value>
            <i4>UserID</i4>
           </value>
           <value>
            <string>Login</string>
           </value>
           <value>
            <i4>AccountID</i4>
           </value>
           <value>
            <string>FName</string>
           </value>
           <value>
            <string>MName</string>
           </value>
           <value>
            <string>LName</string>
           </value>
           <value>
            <string>Email</string>
           </value>
           <value>
            <string>Address1</string>
           </value>
           <value>
            <string>Address2</string>
           </value>
           <value>
            <string>City</string>
           </value>
           <value>
            <string>State</string>
           </value>
           <value>
            <string>Zip</string>
           </value>
           <value>
            <string>CountryID</string>
           </value>
           <value>
            <string>PhCountryCode</string>
           </value>
           <value>
            <string>PhAreaCode</string>
           </value>
           <value>
            <string>PhNumber</string>
           </value>
           <value>
            <string>PhExtention</string>
           </value>
           <value>
            <string>FaxCountryCode</string>
           </value>
           <value>
            <string>FaxAreaCode</string>
           </value>
           <value>
            <string>FaxNumber</string>
           </value>
           <value>
            <string>FaxExtention</string>
           </value>
            <value><i4>0</i4>
           </value>
          </data>
         </array>
        </value>
       </data>
      </array>
     </value>
```

```
    </member>
   </struct>
  </value>
 </param>
</params>
</methodResponse>
```

UserForVendorValidate_API

Description	Parameters	
	Passed	Returned
Method checks if account of user with specified login and password belongs to specified vendor. Method returns ID of account the user belongs to if the account belongs to specified vendor, and "0" if it is not. User password is validated against password filter configured in **System** > **Settings** > **Security** > **Login Settings** submenu of PBA CP. If password does not match the filter, message configured under same menu is returned as second parameter. For example, if customer ID 1000001 belongs to provider (account ID 1), then for all users of this account and vendor account ID 1 the method will return 1000001.	3	3

Syntax

```
ItemResult BM::UserForVendorValidate_API(
  Str(40) StoreLogin;
  Str(40) StorePassword;
  Str(40) StoreVendorID.
)
returns
  Int accountID - ID of account the user belongs to;
  Str passwordStrength - message on weak user password;
  Int UsersID - ID of the user.
```

Special Notes

- **Str(40)** StoreLogin - user login;
- **Str(40)** StorePassword - user password;
- **Str(40)** StoreVendorID - vendor account ID.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all  comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserForVendorValidate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Login -->
         <value>Johnny</value>
         <!-- Password -->
         <value>1q2w3e4r5t</value>
         <!-- Vendor Account ID -->
         <value>1</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
```

```
             <data>
              <!-- Account ID -->
              <value><i4>1000002</i4></value>
              <!-- PasswordStrength -->
              <value><string></string></value>
              <!-- User ID -->
              <value><i4>1000002</i4></value>
             </data>
            </array>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
  </params>
</methodResponse>
```

UserPasswdUpdate_API

Description	Parameters	
	Passed	Returned
Method changes password for specified user. **Note**: method does not propagate information to POA (info on updating the user password will not be transferred to POA).	3	1

Syntax

```
ErrorMessage BM::UserPasswdUpdate_API(
  Int UsersID;
  Str(32) NewPassword;
  Str(32) ConfirmPassword.
)
returns
 Str ErrorMessage -  message after user password has been updated: "Your changes
have been saved."
```

Special Notes

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the Password argument must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42). Avoiding sensitive data is obligatory, otherwise request will fail.

- **Int** UsersID - ID of customer account user. Must exist already in the PBA;
- **Str(32)** NewPassword - new user's password;
- **Str(32)** ConfirmPassword - password confirmation.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserPasswdUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- UsersID -->
         <value><i4>1000002</i4></value>
         <!-- NewPassword -->
         <value>5t4r3e2w1q</value>
         <!-- ConfirmPassword -->
         <value>5t4r3e2w1q</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
```

```
             <member>
              <name>Status</name>
               <value><string>
                Your changes have been saved.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserRemove_API

Description	Parameters	
	Passed	Returned
Method removes user from database.	1	1

Syntax

```
ErrorMessage BM::UserRemove_API(
  Int UsersID.
)
returns
  Str ErrorMessage - message after user has been deleted: "Operation done."
```

Special Notes

Int UsersID - ID of the user to be deleted.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserRemove_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- UsersID -->
         <value><i4>1000026</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserRoleAdd_API

Description	Parameters	
	Passed	Returned
Method assigns an existing role to specified user. Specified role and user must belong to the same account.	2	1

Syntax

```
ErrorMessage BM::UserRoleAdd_API(
  Int UsersID;
  Str RoleName.
)
returns
  Str ErrorMessage - message after role is assigned: "Operation done."
```

Special Notes

Int UsersID - ID of the existing user to assign a role to;

Str RoleName - name of a role that exists in PBA.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserRoleAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- UsersID -->
         <value><i4>1025</i4></value>
         <!-- Role name -->
         <value>Full Access</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
```

```
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserStatusChange_API

	Parameters	
Description	**Passed**	**Returned**
Method enables or disables specified user.	2	1

Syntax

```
ErrorMessage BM::UserStatusChange_API(
  Int UsersID;
  Int NewStatus.
)
returns
  Str ErrorMessage - message after user status has been changed: "Your changes have
been saved."
```

Special Notes

Int UsersID - ID of the user to enabled or disabled;

Int NewStatus - specified user new status: 0 - Enabled, 1 - Disabled.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserStatusChange_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- UsersID -->
         <value><i4>1025</i4></value>
         <!-- New Status -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Your changes have been saved.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserUpdate_API

Description	Parameters	
	Passed	Returned
Method updates user information with given details.	20	1

Important: the method does not propagate information to POA (info on updating the user will not be transferred to POA).

Syntax

```
ErrorMessage BM::UserUpdate_API(
  Int UsersID;
  Str(64) Login;
  Str(30) FName;
  Str(30) MName;
  Str(30) LName;
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention.
)
returns
  Str ErrorMessage -  message after user details have been updated: "Your changes
have been saved."
```

Special Notes

- **Int** UsersID - ID of customer user;
- **Str(64)** Login - customer user login;
- **Str(30)** FName - user contact information: first name;
- **Str(30)** MName - user contact information: middle name;
- **Str(30)** LName - user contact information: last name;
- **Str(100)** Email - user contact information: e-mail address;
- **Str(80)** Address1 - user contact information: address (line 1/2);
- **Str(80)** Address2 - user contact information: address (line 2/2);

- **Str(40)** City - user contact information: city;
- **Str(80)** State - user contact information: state or province;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(4)** PhCountryCode - user contact information: phone country code;
- **Str(10)** PhAreaCode - user contact information: phone area code;
- **Str(20)** PhNumber - user contact information: phone number;
- **Str(10)** PhExtention - user contact information: phone extension;
- **Str(4)** FaxCountryCode - user contact information: fax country code;
- **Str(10)** FaxAreaCode - user contact information: fax area code;
- **Str(20)** FaxNumber - user contact information: fax number;
- **Str(10)** FaxExtention - user contact information: fax extension.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
does not parse xml comments, REMOVE all comments from actual request-->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserUpdate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- UsersID -->
         <value><i4>1000002</i4></value>
         <!-- Login -->
         <value>John Smith</value>
         <!-- FName -->
         <value>John</value>
         <!-- MName -->
         <value>F</value>
         <!-- LName -->
         <value>Smith</value>
         <!-- Email -->
         <value>test@mail.ru</value>
         <!-- Address1 -->
         <value>Moscow</value>
         <!-- Address2 -->
         <value>Moscow</value>
         <!-- City -->
         <value>Moscow</value>
         <!-- State -->
         <value>NW</value>
        <!-- Zip -->
         <value>12345</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PhCountryCode -->
         <value>1</value>
         <!-- PhAreaCode -->
         <value>1234</value>
         <!-- PhNumber -->
         <value>123</value>
         <!-- PhExtention -->
         <value>123</value>
         <!-- FaxCountryCode -->
         <value>1</value>
         <!-- FaxAreaCode -->
         <value>1234</value>
         <!-- FaxNumber -->
         <value>1234</value>
         <!-- FaxExtention -->
         <value>124</value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>
                Your changes have been saved.
              </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserValidate_API

Description	Parameters	
	Passed	Returned
Method returns account and vendor ID of user with specified login.	1	3

Syntax

```
ItemResult BM::UserValidate_API(
  Str Login.
)
returns
  Int AccountID;
  Int VendorID;
  Int UserID.
```

Parameters Desription

Input parameter:

- **Str** Login - user login.

Output parameters:

- **Int** AccountID - identifier of the account to which the user belongs.
- **Int** VendorID - identifier of the vendor associated with the specified user.
- **Int** UserID - identifier of the user.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comment are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all  comments from actual request -- >
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserValidate_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Login -->
         <value>admin</value>
         </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>Result</name>
       <value>
        <array>
         <data>
          <value>
           <array>
            <data>
             <value><i4>1</i4></value>
             <value><i4>-2147483648</i4></value>
             <value><i4>1</i4></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

UserWithIDAdd_API

Description	Parameters	
	Passed	**Returned**
Method creates new user with specified ID. The method is obsolete and left for backward compatibility with PBA versions below .	22	1

Syntax

```
ErrorMessage BM::UserWithIDAdd_API(
  Int AccountID;
  Str(64) Login;
  Str(32) Password;
  Str(32) ConfirmPassword;
  Str(30) FName;
  Str(30) LName;
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention;
  Int UsersID.
)
returns
 Str ErrorMessage -  message after user has been added: "Operation done."
```

Special Notes

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the Password and ConfirmPassword arguments must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42). Avoiding sensitive data is obligatory, otherwise request will fail.

- **Int** AccountID - customer account ID;
- **Str(64)** Login - customer user login;
- **Str(32)** Password - user's password;
- **Str(32)** ConfirmPassword - password confirmation field;

- **Str(30)** FName - user contact information: first name;
- **Str(30)** MName - user contact information: middle name;
- **Str(30)** LName - user contact information: last name;
- **Str(100)** Email - user contact information: e-mail address;
- **Str(80)** Address1 - user contact information: address (line 1/2);
- **Str(80)** Address2 - user contact information: address (line 2/2);
- **Str(40)** City - user contact information: city;
- **Str(80)** State - user contact information: state or province;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(4)** PhCountryCode - user contact information: phone country code;
- **Str(10)** PhAreaCode - user contact information: phone area code;
- **Str(20)** PhNumber - user contact information: phone number;
- **Str(10)** PhExtention - user contact information: phone extension;
- **Str(4)** FaxCountryCode - user contact information: fax country code;
- **Str(10)** FaxAreaCode - user contact information: fax area code;
- **Str(20)** FaxNumber - user contact information: fax number;
- **Str(10)** FaxExtention - user contact information: fax extension;
- **Int** UsersID - ID of customer user.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserWithIDAdd_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
         <!-- Login -->
         <value>admin</value>
         <!-- Password -->
         <value>setup</value>
         <!-- ConfirmPassword -->
         <value>setup</value>
         <!-- FName -->
         <value>John</value>
         <!-- LName -->
         <value>Smith</value>
         <!-- Email -->
         <value>test@mail.ru</value>
         <!-- Address1 -->
         <value>Moscow</value>
         <!-- Address2 -->
         <value>Moscow</value>
         <!-- City -->
         <value>Moscow</value>
         <!-- State -->
         <value>NW</value>
         <!-- Zip -->
         <value>12345</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PhCountryCode -->
         <value>1</value>
         <!-- PhAreaCode -->
         <value>1234</value>
         <!-- PhNumber -->
         <value>123</value>
         <!-- PhExtention -->
         <value>123</value>
         <!-- FaxCountryCode -->
         <value>1</value>
         <!-- FaxAreaCode -->
         <value>1234</value>
         <!-- FaxNumber -->
         <value>1234</value>
         <!-- FaxExtention -->
         <value>124</value>
         <!-- UsersID -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

UserWithIDAddWithRole_API

Description	Parameters	
	Passed	**Returned**
Method creates new user with specified ID.	23	1

The method does not call POA to create user, for that purpose UserAdd_API (on page 633) should be used.

Syntax

```
ErrorMessage BM::UserWithIDAddWithRole_API(
  Int AccountID;
  Str(64) Login;
  Str(32) Password;
  Str(32) ConfirmPassword;
  Str(30) FName;
  Str(30) LName;
  Str(100) Email;
  Str(80) Address1;
  Str(80) Address2;
  Str(40) City;
  Str(80) State;
  Str(10) Zip;
  Str(2) CountryID;
  Str(4) PhCountryCode;
  Str(10) PhAreaCode;
  Str(20) PhNumber;
  Str(10) PhExtention;
  Str(4) FaxCountryCode;
  Str(10) FaxAreaCode;
  Str(20) FaxNumber;
  Str(10) FaxExtention;
  Int UsersID;
  Int AddFARole.
)
returns
 Str ErrorMessage -  message after user has been added: "Operation done."
```

Special Notes

Important: Sensitive data is passed to the method. To avoid logging of sensitive data, the Password and ConfirmPassword arguments must be escaped as this described in the section Important: Avoid Sensitive Data Logging (on page 42). Avoiding sensitive data is obligatory, otherwise request will fail.

- **Int** AccountID - customer account ID;
- **Str(64)** Login - customer user login;
- **Str(32)** Password - user's password;

- **Str(32)** ConfirmPassword - password confirmation field;
- **Str(30)** FName - user contact information: first name;
- **Str(30)** MName - user contact information: middle name;
- **Str(30)** LName - user contact information: last name;
- **Str(100)** Email - user contact information: e-mail address;
- **Str(80)** Address1 - user contact information: address (line 1/2);
- **Str(80)** Address2 - user contact information: address (line 2/2);
- **Str(40)** City - user contact information: city;
- **Str(80)** State - user contact information: state or province;
- **Str(10)** Zip - user contact information: zip or postal code. It will be verified through country specific regular expression. You could modify regular expression under countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(2)** CountryID - user contact information: two-letter country code, verified against countries list in **System** > **Settings** > **Internationalization** > **Countries** submenu of the Navigation tree;
- **Str(4)** PhCountryCode - user contact information: phone country code;
- **Str(10)** PhAreaCode - user contact information: phone area code;
- **Str(20)** PhNumber - user contact information: phone number;
- **Str(10)** PhExtention - user contact information: phone extension;
- **Str(4)** FaxCountryCode - user contact information: fax country code;
- **Str(10)** FaxAreaCode - user contact information: fax area code;
- **Str(20)** FaxNumber - user contact information: fax number;
- **Str(10)** FaxExtention - user contact information: fax extension;
- **Int** UsersID - ID of customer user;
- **Int** AddFARole - whether Full Access role is assigned to created user (0 - No, 1 - Yes).

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>UserWithIDAddWithRole_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- AccountID -->
         <value><i4>123</i4></value>
         <!-- Login -->
         <value>admin</value>
         <!-- Password -->
         <value>setup</value>
         <!-- ConfirmPassword -->
         <value>setup</value>
         <!-- FName -->
         <value>John</value>
         <!-- LName -->
         <value>Smith</value>
         <!-- Email -->
         <value>test@mail.ru</value>
         <!-- Address1 -->
         <value>Moscow</value>
         <!-- Address2 -->
         <value>Moscow</value>
         <!-- City -->
         <value>Moscow</value>
         <!-- State -->
         <value>NW</value>
         <!-- Zip -->
         <value>12345</value>
         <!-- CountryID -->
         <value>us</value>
         <!-- PhCountryCode -->
         <value>1</value>
         <!-- PhAreaCode -->
         <value>1234</value>
         <!-- PhNumber -->
         <value>123</value>
         <!-- PhExtention -->
         <value>123</value>
         <!-- FaxCountryCode -->
         <value>1</value>
         <!-- FaxAreaCode -->
         <value>1234</value>
         <!-- FaxNumber -->
         <value>1234</value>
         <!-- FaxExtention -->
         <value>124</value>
         <!-- UsersID -->
         <value><i4>1032</i4></value>
         <!-- AddFARole -->
         <value><i4>1</i4></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <struct>
             <member>
              <name>Status</name>
               <value><string>Operation done. </string></value>
             </member>
            </struct>
           </value>
          </data>
         </array>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
  </methodResponse>
```

ValidatePasswordQuality_API

Description	Parameters	
	Passed	**Returned**
Method validates the quality of submitted password.	1 (+ 4 optional)	1

The password string is validated in accordance with the password quality level of the Provider against the following parameters:

- Login name.
- User first and last names (Gecos).
- VendorID.
- OldPassword.
- Five previous passwords for user (Determined by the login name).

Syntax

```
ItemResult BM::ValidatePasswordQuality_API(
  Str Password;
  Str Login;
  Str Gecos;
  Int VendorID;
  Str OldPassword;
)
returns
  Str Message.
```

Parameters description

Input parameters:

- **Str** Password - the password string to be validated. Mandatory parameter.
- **Str** Login - existing user login; the check that password is not based on the login name. Optional.
- **Str** Gecos - the user's first and last names. Optional parameter.
- **Int** VendorID - ID of the vendor from the point of which the password is validated. Different vendors may have different settings of password quality level. Optional parameter; if not specified, the Provider ID is used.
- **Str** OldPassword - the old password against which the new one is validated; the check that the new password is not based on an old one. The parameter is optional.

Note: If you specified only NewPassword, then it will be validated against the Provider's password quality level.

Output parameters:

- **Str** Message - validation result message.

Example

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Warning! Comments are only given for you convenience. PBA XMLRPC
 does not parse xml comments, REMOVE all comments from actual request -->
<methodCall>
 <methodName>Execute</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Server</name>
      <value>BM</value>
     </member>
     <member>
      <name>Method</name>
      <value>ValidatePasswordQuality_API</value>
     </member>
     <member>
      <name>Params</name>
      <value>
       <array>
        <data>
         <!-- Password -->
         <value>
                <string>1q2w3e$r_Iamadmin</string>
              </value>
         <!-- Login -->
         <value>
                <string>admin</string>
              </value>
         <!-- Gecos -->
         <value>
                <string>John F. Smith</string>
              </value>
         <!-- VendorID -->
         <value><i4>1001</i4></value>
         <!-- OldPassword -->
              <value><string>1q2w3e$r_Iam</string></value>
        </data>
       </array>
      </value>
     </member>
    </struct>
   </value>
  </param>
 </params>
</methodCall>
```

Response

```
<?xml version="1.0"?>
 <methodResponse>
  <params>
```

```
   <param>
   <value>
    <struct>
      <member>
       <name>Result</name>
        <value>
         <array>
          <data>
           <value>
            <array>
             <data>
              <name>Status</name>
               <value><string>
                                <!-- Empty string in case of successful validation -->
             </string></value>
            </data>
           </array>
          </value>
         </data>
        </array>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
 </methodResponse>
```

Index