



Xeround API Service Reference

API Version 1.1

February, 2012

Table of Contents

Introduction	1
Recent changes	2
Actions	2
Common errors	2
Actions	3
createDbInstance	3
Get - CreateDbInstanceRequest	3
Response - NewDbInstanceInfo	5
Throw - XeroundApiFault	5
Examples	6
dropDbInstance	8
Get - DropDbInstanceRequest	8
Throw - XeroundApiFault	8
Examples	8
getDbInstanceInfo	9
Get - GetDbInstanceInfoRequest	9
Response – DbInstanceInfo[]	9
Throw - XeroundApiFault	11
Examples	11
changeDbInstancePassword	14
Get - SetUserPasswordRequest	14
Throw - XeroundApiFault	14
Examples	14
Error handling	16
XeroundApiFault	16
Contents	16
FaultInfo	16
Example for FAULT	16
Example for working with Java using xeround-api-service-1.0.jar	18
SoapClientTest.java	18

ClientPasswordCallback.java.....	20
REST Example in Java	20
RestClientTest.java.....	20

Introduction

The Xeround Service API allows developers to connect third-party applications to Xeround service and automate processes available in the UI. The API is accessible using either REST or SOAP, per your preference.

To use the API you need to be a registered user of the service. The connection is secured over HTTPS, authentication is done with your user credentials added to the HTTP header via HTTP Basic Authentication.

To access the API's landing page navigate with your browser to: <https://api.xeround.com:8443>

To access the API using SOAP: <https://api.xeround.com:8443/1.1/soap?wsdl>

To access the API using REST: <https://api.xeround.com:8443/1.1/rest>

To use from a JAVA application use `xeround-api-service-1.0.jar` that can be downloaded freely from the following URL: <http://xeround.com/api/xeround-api-service-1.0.jar>

Note: the API uses a certificate issued by GoDaddy. You may need to add it to your trusted certificate issuers (see the following instructions) and make sure that your application uses that keystore:

- 1) Access <https://certs.godaddy.com/anonymous/repository.seam>
- 2) Download the Go Daddy Secure Server Certificate (Intermediate Certificate) - `gd_intermediate.crt`
- 3) Import GoDaddy's certificate into the trust store using the following command:

```
./keytool -import -alias "gd_intermediate" -file  
gd_intermediate.crt -keystore xeround.service.api.truststore
```

Recent changes

Xeround Cloud Database is available now in three editions, providing you the flexibility to choose the most suitable edition for you:

- Xeround FREE – up to 10MB
- Xeround BASIC – up to 250MB
- Xeround PRO – up to 50GB

The new version of our Service API v1.1 includes a reference to Xeround editions
CreateDbInstanceRequest & DbInstanceInfo must include now the desired edition for the instance.

For information about the various editions and prices please review <http://xeround.com/mysql-cloud-db-overview/pricing>

Actions

The following actions are currently supported:

- createDbInstance
- dropDbInstance
- getDbInstanceInfo
- changeDbInstancePassword

Common errors

There are 2 common errors that all actions return:

Code	Description
201	Unable to perform the action
202	Invalid parameter value

For action-specific errors please see details in the topic for the action.

Actions

createDbInstance

Description Creates a new DB Instance.

Note: Instance creation might take few minutes. You should query the instance status using `getDbInstanceInfo` to verify it was successfully created and to get its DNS name.

Get - CreateDbInstanceRequest

Name	Description	Type	Required
name	<p>The DB Instance name is used to access the instance and to manage it via the DB Instance Manager UI.</p> <p>Len <=45</p> <p>Unique for user</p> <p>Characters = a-z A-Z 0-9 _ -</p>	String	Yes
initialSize	<p>The initial amount of memory to be allocated to a PRO DB Instance. Specified in GB.</p> <p>This value must be NULL for FREE and BASIC instances as their size is fixed.</p> <p>Xeround is fully scalable and elastic; hence resources allocation will be adjusted (both up and down) according to the DB Instance's actual size. Nevertheless, providing a good estimate of the initial size will optimize resource usage during initial load. Note that if you choose a larger initial size than the actual data you import to Xeround, your DB Instance will automatically shrink to its actual size a week after its creation.</p> <p>Valid values: On Amazon: minimum 0.5GB maximum 5GB</p>	Float	No

	<p>On Rackspace: minimum 0.5GB maximum 2.5GB</p> <p>Default: Amazon 0.5 GB, Rackspace 0.5GB</p>		
username	<p>MySQL username to access the DB Instance</p> <p>Constraints: length <= 16, characters - reserved word - root and xeround_</p>	String	Yes
password	<p>MySQL password to access the DB Instance</p> <p>Constraints: length <= 42</p>	String	Yes
dailyBackupTime	<p>The desired time window for the daily backup run. Note: this parameter is relevant for Xeround PRO only. Should be NULL for other plan.</p> <p>Valid values : 0 - 23</p> <p>Default: A random, system-chosen time</p>	Integer	No
description	<p>A textual description related to the DB Instance</p> <p>Constraints: length < = 256</p>	String	No
dataCenter	<p>The physical datacenter where the DB Instance is hosted.</p> <p>Valid values: Amazon-EC2-US-East Amazon-EC2-EU-West Rackspace-US-ORD1</p>	String	Yes
edition	<p>The desired DB instance plan/edition</p> <p>Valid values: FREE (up to 10MB) BASIC (up to 250MB) PRO (different size limits apply to each data center, as mentioned in the description of the initialSize parameter)</p> <p>Default: PRO</p>	String	Yes

Response - NewDbInstanceInfo

Name	Description	Type
dbInstanceId	A unique ID that identifies the DB Instance. This ID should be provided in the relevant API calls	int
plan	The billing plan used to calculate the DB Instance cost Possible values: <ul style="list-style-type: none"> • Paid • Trial • Free Null means the data center is non-billable	String

Throw - XeroundApiFault

Error Code	Description	Code
DBInstanceCreationFailed	Unable to create DB instance	201
InvalidDBInstanceNameBlank	DB Instance name cannot be blank	202
InvalidDBInstanceNameLength	DB Instance name length cannot be more than 45 characters	202
DBInstanceNameAlreadyExists	User already has a DB Instance with the given name	202
InvalidDBInstanceName	The specified DB Instance name contains invalid characters. Valid characters are a-z A-Z 0-9 _ -	202
DBInstanceInitialSizeTooBig	Requested DB Instance size {InitialSize}GB is greater than max allowed initial size {MaxInitialSize}GB	202
DBInstanceInitialSizeTooSmall	Requested DB Instance size {InitialSize}GB is smaller than min allowed initial size {MinInitialSize}GB	202
InvalidUsernameBlank	Username cannot be blank	202
InvalidUsernameLength	Username length cannot be more than 16 characters	202
InvalidUsernameReserved	Requested username is a reserved word - root or xeround_	202
InvalidPasswordBlank	Password cannot be blank	202
InvalidPasswordLength	Password length cannot be more than 42 characters	202
InvalidBackupTime	Daily Backup Time can be provided only with a PRO instance and must be a value between 0 and 23	202
InvalidDescriptionLength	Description length cannot be more than 256 characters	
InvalidDataCenterBlank	Data Center name cannot be blank	202
InvalidDataCenter	Specified Data Center does not exist	202

DataCenterNotAllowed	DataCenter {DataCenter} is not among the allowed data centers {AllowedDataCenters} for account	202
DBInstanceQuotaExceeds	Max number of instances exceeded for account .	202
NoPaymentDetails	The account does not have entered payment details	202
InvalidDefaultPaymentDetails	Default payment method must be a credit card	202
InvalidPaymentDetailsExpired	Credit Card expired at {month/year}	202
DataCenterQuotaExceeds	Due to high demand on {DataCenter}, the request is denied. Please select a different data center or try again later	203
DBInstanceInitialSizeMustBeNull	DB Instance size for {edition} must be null.	202

Examples

SOAP Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soapenv:Header/>
  <soapenv:Body>
    <api:createDbInstance>
      <api:createDbInstanceRequest>
        <api:dailyBackupTime>0</api:dailyBackupTime>
        <api:dataCenter>Amazon-EC2-US-East</api:dataCenter>
        <api:description>MyDbInstance description</api:description>
        <api:edition>PRO</api:edition>
        <api:initialSize>1</api:initialSize>
        <api:name>MyDbInstance</api:name>
        <api:password>MyPassword</api:password>
        <api:username>MyUsername</api:username>
      </api:createDbInstanceRequest>
    </api:createDbInstance>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soap:Body>
    <api:createDbInstanceResponse>
      <return>
        <dbInstanceId>{DbInstanceId}</dbInstanceId>
        <plan>Trial</plan>
      </return>
    </api:createDbInstanceResponse>
  </soap:Body>
</soap:Envelope>
```

```
</soap:Body>  
</soap:Envelope>
```

REST Request

```
POST https://api.xeround.com:8443/1.1/rest/db_instances  
  
<?xml version="1.0" encoding="UTF-8"?>  
<CreateDbInstanceRequest xmlns="http://api.xeround.com">  
  <name>MyDbInstance</name>  
  <initialSize>1</initialSize>  
  <username>MyUsername</username>  
  <password>MyPassword</password>  
  <dailyBackupTime>1</dailyBackupTime>  
  <description>MyDbInstance description</description>  
  <dataCenter>Amazon-EC2-US-East</dataCenter>  
  <edition>PRO</edition>  
</CreateDbInstanceRequest>
```

REST Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<NewDbInstanceInfo xmlns="http://api.xeround.com">  
  <dbInstanceId>{DbInstanceId}</dbInstanceId>  
  <plan>Trial</plan>  
</NewDbInstanceInfo>
```

dropDbInstance

Description Drops an existing DB Instance.

Get - DropDbInstanceRequest

Name	Description	Type	Required
dbInstanceid	The unique ID that identifies the DB Instance	Integer	Yes

Throw - XeroundApiFault

ErrorCode	Description	Code
DBInstanceDropFailed	Unable to drop DB instance	201
InvalidDBInstanceBlank	DB Instance Id cannot be blank	202
DBInstanceNotFound	Specified DB Instance does not exists	202

Examples

SOAP Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soapenv:Header/>
  <soapenv:Body>
    <api:dropDbInstance>
      <api:dropDbInstanceRequest>
        <api:dbInstanceid>{DbInstanceid}</api:dbInstanceid>
      </api:dropDbInstanceRequest>
    </api:dropDbInstance>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soapenv:Header/>
  <soapenv:Body>
    <api:dropDbInstance>
      <api:dropDbInstanceRequest>
      </api:dropDbInstanceRequest>
    </api:dropDbInstance>
  </soapenv:Body>
</soapenv:Envelope>
```

REST Request

```
DELETE https://api.xeround.com:8443/1.1/rest/db\_instances/{DbInstanceid}
```

getDbInstanceInfo

Description Gets all relevant information for an existing DB Instance or for all instances of a specific account

Get - GetDbInstanceInfoRequest

Name	Description	Type	Required
dbInstanceid	The unique ID that identifies the DB Instance If not specified - returns a list of all of the user active instances with their information	Integer	No

Response - DbInstanceInfo[]

Name	Description	Type
dbInstanceid	The unique ID that identifies the DB Instance	int
status	Specifies the current status/state of this database. Health status: HEALTHY NO_CONNECTIVITY CRITICAL Static and dynamic states: ACTIVE STOPPED DROPPED FAILED ALLOCATING INITIALIZING STARTING LOADING STOPPING RESTORING UPGRADING SCALE_UP SCALE_DOWN CHANGING_PLAN RELOCATING For more details about the above statuses please see Xeround User Guide	String

percentage	Specifies the progress of progressive statuses such as STARTING, STOPPING, RESTORING,.... Set to 0 when not relevant.	Integer
size	Specifies the actual memory size. Specified in GB.	Float
name	Contains the name of the DB Instance	String
description	Contains the textual description entered for the DB Instance	String
dataCenter	The physical datacenter where the DB Instance is hosted. Valid values: Amazon-EC2-US-East Amazon-EC2-EU-West Rackspace-US-ORD1	String
dnsName	The DNS name (Round Robin) that should be used to access the DB Instance from an external data center.	String
internalDnsName	DNS name (Round Robin) when accessing from within the same datacenter.	String
accessPoints	List of all valid IP's of your instance front end access points. These IP's can be used to access the instance from an external data center in case the DNS is not active.	AccessPointsEntry[]
internalAccessPoints	List of all valid internal IP's of your instance front end access points. These IPs can be used to access the instance from within the same data center, in case the DNS is not active	AccessPointsEntry[]*
Plan	The billing plan used to calculate the DB Instance cost Possible values: <ul style="list-style-type: none"> • Paid • Trial • Free Null means the data center the is non-billable	String
remainingPlanDays	Specifies the number of remaining days for current billing plan. For example – if the instance is in free	Integer

	<p>trial period this parameter will specify the remaining days for the trial.</p> <p>Null means the billing plan is not limited by time</p>	
creationDate	The date the DB Instance was created	Date
edition	The DB Instance type/edition. i.e. FREE, BASIC, PRO.	String

AccessPointsEntry

Description contains a list of the valid access points / IP's to access the DB Instance. This data type is used in the response of getDbInstanceInfo

Contents

Name	Description	Type
dataCenter	<p>The physical datacenter where the DB Instance is hosted.</p> <p>Valid values: Amazon-EC2-US-East Amazon-EC2-EU-West Rackspace-US-ORD1</p>	String
accessPoints	List of all valid IP's that can be used to access the instance.	String[]

Throw - XeroundApiFault

ErrorCode	Description	Code
GetDBInstanceInfoFailed	Unable to get DB instance info	201
DBInstanceNotFound	Specified DB Instance does not exists	202

Examples

SOAP Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
<soapenv:Header/>
<soapenv:Body>
<api:getDbInstanceInfo>
<api:getDbInstanceInfoRequest>
<api:dbInstanceId>{DbInstanceId}</api:dbInstanceId>
</api:getDbInstanceInfoRequest>
</api:getDbInstanceInfo>
</soapenv:Body>
```

```
</soapenv:Envelope>
```

SOAP Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soap:Body>
    <api:getDbInstanceInfoResponse>
      <return>
        <accessPoints>
          <accessPoints>dev6.xeround.com:3004</accessPoints>
          <accessPoints>dev9.xeround.com:3004</accessPoints>
          <accessPoints>dev7.xeround.com:3004</accessPoints>
          <dataCenter>Amazon-EC2-US-East</dataCenter>
        </accessPoints>
        <creationDate>2011-09-18T23:54:33+03:00</creationDate>
        <dataCenter>Amazon-EC2-US-East</dataCenter>
        <dbInstanceid>{DbInstanceid}</dbInstanceid>
        <edition>PRO</edition>
        <description>MyDbInstance Description</description>
        <name>MyDbInstance</name>
        <plan>Trial</plan>
        <remainingPlanDays>16</remainingPlanDays>
        <size>1</size>
        <status>HEALTHY</status>
      </return>
    </api:getDbInstanceInfoResponse>
  </soap:Body>
</soap:Envelope>
```

REST Request

```
GET https://api.xeround.com:8443/1.1/rest/db\_instances/{DbInstanceid}/info
GET https://api.xeround.com:8443/1.1/rest/db\_instances/info
```

REST Response

```
<ns1:DbInstanceInfos xmlns:ns1="http://api.xeround.com">
  <ns1:DbInstanceInfo xmlns:ns1="http://api.xeround.com">
    <ns1:accessPoints>
      <ns1:accessPoints>dev6.xeround.com:3008</ns1:accessPoints>
      <ns1:accessPoints>dev7.xeround.com:3008</ns1:accessPoints>
      <ns1:accessPoints>dev9.xeround.com:3008</ns1:accessPoints>
      <ns1:dataCenter>Amazon-EC2-US-East</ns1:dataCenter>
    </ns1:accessPoints>
    <ns1:creationDate>2011-09-18T23:26:25+03:00</ns1:creationDate>
    <ns1:dataCenter>Amazon-EC2-US-East</ns1:dataCenter>
    <ns1:dbInstanceid>{DbInstanceid}</ns1:dbInstanceid>
    <ns1:description>MyDbInstance description</ns1:description>
```

```
<ns1:edition>PRO</ns1:edition>  
<ns1:name>MyDbInstance</ns1:name>  
<ns1:plan>Trial</ns1:plan>  
<ns1:remainingPlanDays>16</ns1:remainingPlanDays>  
<ns1:size>1</ns1:size>  
<ns1:status>HEALTHY</ns1:status>  
</ns1:DbInstanceInfo>  
</ns1:DbInstanceInfos>
```

changeDbInstancePassword

Description changes the MySQL password to access a DB Instance.

Get - SetUserPasswordRequest

Name	Description	Type	Required
dbInstanceId	The unique ID that identifies the DB Instance	Integer	Yes
newPassword	The new MySQL password to access the DB Instance	String	Yes

Throw - XeroundApiFault

ErrorCode	Description	Code
ChangePasswordFailed	Unable to change DB instance password	201
InvalidDBInstanceBlank	DB Instance Id cannot be blank	202
InvalidNewPasswordBlank	Specified new password cannot be blank	202
DBInstanceNotFound	Specified DB Instance does not exist	202

Examples

SOAP Request

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soapenv:Header/>
  <soapenv:Body>
    <api:changeDbInstancePassword>
      <api:changeDbInstancePasswordRequest>
        <api:dbInstanceId>{DbInstanceId}</api: dbInstanceId>
        <api:newPassword>MyNewPassword</api:newPassword>
      </api:changeDbInstancePasswordRequest>
    </api:changeDbInstancePassword>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soapenv:Header/>
  <soapenv:Body>
    <api:changeDbInstancePassword>
      <api:changeDbInstancePasswordRequest>
```

```
</api:changeDbInstancePasswordRequest>  
</api:changeDbInstancePassword>  
</soapenv:Body>  
</soapenv:Envelope>
```

REST Request

```
POST https://api.xeround.com:8443/1.1/rest/db\_instances/{DbInstanceid}/password  
  
<?xml version="1.0" encoding="UTF-8"?>  
  <ChangeDbInstancePasswordRequest xmlns="http://api.xeround.com">  
    <newPassword>MyNewPassword</newPassword>  
  </ChangeDbInstancePasswordRequest>
```

Error handling

XeroundApiFault

Description Contains the FaultInfo data type which holds details for of a failed action.

Contents

Name	Description	Type
faultInfo		FaultInfo

FaultInfo

Description contains details for a failed action
This data type is used as an element in the XeroundApiFault data type

Contents

Name	Description	Type
description		String
errorCode		String
code		int
faultId	Service fault code	String
requestId	Request's unique id	String

Example for FAULT

SOAP

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:api="http://api.xeround.com">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Fault occurred while processing.</faultstring>
      <detail>
        <XeroundApiFault xmlns="http://api.xeround.com">
          <description>Name 'MyDbInstance' is already in use</description>
          <errorCode>202</errorCode>
          <requestId>20110918234744017001</requestId>
        </XeroundApiFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

```
</soap:Fault>  
</soap:Body>  
</soap:Envelope>
```

REST

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<FaultInfo xmlns="http://api.xeround.com">  
  <description>Name 'MyDbInstance' is already in use</description>  
  <errorCode>202</errorCode>  
  <requestId>20110918230113500006</requestId>  
</FaultInfo>
```

Example for working with Java using xeround-api-service-1.1.jar

SoapClientTest.java

```
public final class SoapClientTest {

    private static final QName SERVICE_NAME = new
QName("http://api.xeround.com", "XeroundApiService");

    private SoapClientTest() {
    }

    public static void main(String args[]) throws Exception {

        URL wsdlURL = XeroundApiService.WSDL_LOCATION;

        XeroundApiService ss = new XeroundApiService(wsdlURL, SERVICE_NAME);
        XeroundApi port = ss.getXeroundApiPort();

        Client client = ClientProxy.getClient(port);
        Endpoint cxfEndpoint = client.getEndpoint();

        Map<String, Object> outProps = new HashMap<String, Object>();
        WSS4JOutInterceptor wssOut = new WSS4JOutInterceptor(outProps);
        cxfEndpoint.getOutInterceptors().add(wssOut);

        outProps.put(WSHandlerConstants.ACTION,
WSHandlerConstants.USERNAME_TOKEN);
        outProps.put(WSHandlerConstants.USER, "username");
        outProps.put(WSHandlerConstants.PASSWORD_TYPE, WSConstants.PW_TEXT);
        outProps.put(WSHandlerConstants.PW_CALLBACK_CLASS,
ClientPasswordCallback.class.getName());

        try {
            System.out.println("Invoking createDbInstance...");
            CreateDbInstanceRequest createDbInstanceRequest = new
CreateDbInstanceRequest();
            createDbInstanceRequest.setDataCenter("Amazon-EC2-US-East");
            createDbInstanceRequest.setName("MyDbInstance");
            createDbInstanceRequest.setDescription("MyDbInstance
Description");
            createDbInstanceRequest.setDailyBackupTime(1);
            createDbInstanceRequest.setInitialSize(1f);
            createDbInstanceRequest.setUsername("MyUsername");
            createDbInstanceRequest.setPassword("MyPassword");

            // Sending the request and getting the basic information on the
instance being created
            NewDbInstanceInfo createDbInstanceResponse =
port.createDbInstance(createDbInstanceRequest);
            int instanceId = createDbInstanceResponse.getDbInstanceId();
            System.out.println("createDbInstance.result=" + instanceId + ",
plan=" + createDbInstanceResponse.getPlan());
        }
    }
}
```

```
String status;
do {
    Thread.sleep(5000);

    System.out.println("Invoking getDbInstanceInfo...");
    GetDbInstanceInfoRequest getDbInstanceInfoRequest = new
getDbInstanceInfoRequest();

    getDbInstanceInfoRequest.setDbInstanceId(instanceId);

    List<DbInstanceInfo> getDbInstanceInfoResponse =
port.getDbInstanceInfo(getDbInstanceInfoRequest);
    status = getDbInstanceInfoResponse.get(0).getStatus();

    System.out.println("getDbInstanceInfo.response: Current
instance status = " + status);

    // while status is initializing or allocating
} while (status.equals("INITIALIZING") ||
status.equals("ALLOCATING"));

    System.out.println("Instance " + instanceId + " created, current
status = " + status);

    // Getting information for all the instances
    System.out.println("Invoking getDbInstanceInfo for all
instances...");
    GetDbInstanceInfoRequest getDbInstanceInfoRequest = new
getDbInstanceInfoRequest();
    List<DbInstanceInfo> getDbInstanceInfoResponse =
port.getDbInstanceInfo(getDbInstanceInfoRequest);
    System.out.println("getDbInstanceInfo.response: Total instances
count " + getDbInstanceInfoResponse.size());

    // Changing the database password for user "test" - the username
used in the creation request
    System.out.println("Invoking changeDbInstancePassword...");
    ChangeDbInstancePasswordRequest changeDbInstancePasswordRequest =
new ChangeDbInstancePasswordRequest();
    changeDbInstancePasswordRequest.setDbInstanceId(instanceId);
    changeDbInstancePasswordRequest.setNewPassword("MyNewPaswword");
    port.changeDbInstancePassword(changeDbInstancePasswordRequest);

    // Dropping the database instance
    System.out.println("Invoking dropDbInstance...");
    DropDbInstanceRequest dropDbInstanceRequest = new
DropDbInstanceRequest();
    dropDbInstanceRequest.setDbInstanceId(instanceId);
    port.dropDbInstance(dropDbInstanceRequest);

    System.out.println("SUCCESS");

} catch (XeroundApiFault e) {
```

```
        System.out.println("Xeround Service Exception: '" +
e.getFaultInfo().getDescription() + "'");
        e.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

ClientPasswordCallback.java

```
import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import org.apache.ws.security.WSPasswordCallback;

public class ClientPasswordCallback implements CallbackHandler {

    @Override
    public void handle(Callback[] callbacks) throws IOException,
UnsupportedCallbackException {

        WSPasswordCallback pc = (WSPasswordCallback) callbacks[0];

        // set the password for our message.
        pc.setPassword("password");
    }
}
```

REST Example in Java

RestClientTest.java

```
public class RestClientTest {

    private static final String USERNAME = "username";
    private static final String PASSWORD = "password";
    private static final String NS1_DB_INSTANCE_ID_START = "<dbInstanceId>";
    private static final String NS1_DB_INSTANCE_ID_END = "</dbInstanceId>";
    private static final String CREATE_DB_INSTANCE_XML = "<?xml
version='1.0' encoding='UTF-8'>\n"+
        "<CreateDbInstanceRequest xmlns='http://api.xeround.com'>\n"+
        "\t<name>MyDbInstance</name>\n"+
        "\t<initialSize>1</initialSize>\n"+
        "\t<dataCenter>Amazon-EC2-US-East</dataCenter>\n"+
        "\t<username>MyUsername</username>\n"+
        "\t<password>MyPassword</password>\n"+
        "\t<dailyBackupTime>1</dailyBackupTime>\n"+
        "\t<description>MyDbInstance description</description>\n"+
        "</CreateDbInstanceRequest>";
    private static final String CHANGE_DB_PASSWORD_XML = "<?xml
version='1.0' encoding='UTF-8'>\n"+
```

```
        "<ChangeDbInstancePasswordRequest
xmlns=\"http://api.xeround.com\">\n"+
        "\t<newPassword>MyNewPassword</newPassword>\n"+
        "</ChangeDbInstancePasswordRequest>";

    public static void main(String[] args) throws Exception {

        String response;

        String restUrlPrefix = "https://api.xeround.com:8443/1.0/rest";
        if (args.length > 0) {
            restUrlPrefix = args[0];
        }
        // Sending the request and getting the basic information on the
instance being created
        response = request(restUrlPrefix + "/db_instances", "POST",
CREATE_DB_INSTANCE_XML);
        System.out.println("Create response = " +
validateResponse(response));

        int instanceId =
Integer.parseInt(response.substring(response.indexOf(NS1_DB_INSTANCE_ID_START
) + NS1_DB_INSTANCE_ID_START.length(),
response.indexOf(NS1_DB_INSTANCE_ID_END)));
        do {
            Thread.sleep(5000);

            response = request(restUrlPrefix + "/db_instances/" + instanceId
+ "/info", "GET", null);
            System.out.println("GetInfo response = " +
validateResponse(response));

            // while status is initializing or allocating
        } while (response.contains("<ns1:status>INITIALIZING</ns1:status>")
|| response.contains("<ns1:status>ALLOCATING</ns1:status>"));

        // Getting information for all the instances
        response = request(restUrlPrefix + "/db_instances/info", "GET",
null);
        System.out.println("Get response = " + validateResponse(response));

        // Changing the database password for user "test" - the username used
in the creation request
        response = request(restUrlPrefix + "/db_instances/" + instanceId +
"/password", "POST", CHANGE_DB_PASSWORD_XML);
        System.out.println("GetAll response = " +
validateResponse(response));

        // Dropping the database instance
        response = request(restUrlPrefix + "/db_instances/" + instanceId,
"DELETE", null);
        System.out.println("Drop response = " + validateResponse(response));

        System.out.println("SUCCESS");
    }
}
```

```
private static String validateResponse(String response) {
    if (response.contains("<FaultInfo ") {
        throw new RuntimeException(response);
    }
    return response;
}

private static String request(String uri, String method, String content)
throws Exception {

    HttpURLConnection conn = null;
    OutputStreamWriter wr = null;
    BufferedReader in = null;
    String response = "";

    try {
        URL url = new URL(uri);
        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod(method);
        setAuthorization(conn);
        if (content != null) {
            conn.setDoOutput(true);
            conn.setRequestProperty("Content-Type", "application/xml");
            wr = new OutputStreamWriter(conn.getOutputStream());
            wr.write(content);
            wr.flush();
        }
        in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        String responseLine;
        while ((responseLine = in.readLine()) != null) {
            response += responseLine + "\n";
        }
    } finally {
        closeResources(conn, wr, in);
    }

    return response;
}

private static void setAuthorization(HttpURLConnection conn) {
    BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedAuthorization = enc.encode((USERNAME + ":" +
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " +
encodedAuthorization);
}

private static void closeResources(HttpURLConnection conn,
OutputStreamWriter wr, BufferedReader in) {
    try {
        if (wr != null) {
            wr.close();
        }
    }
}
```

```
        if (in != null) {
            in.close();
        }
        if (conn != null) {
            conn.disconnect();
        }
    } catch (Exception e) {
        System.out.println("Could not close HttpURLConnection " + e);
    }
}
}
```