WebSMS Digi API

- Updated 2020-06 with priority, all error codes, new method sendSmsExtended, callback

Host : https://smsapi.rcs-rds.ro/jsonrpc/public

Protocol : HTTP(JSON-RPC)

Authentication: Basic Auth

Endpoints :

- My Account
  - **URL : https://smsapi.rcs-rds.ro/jsonrpc/public/my-account**
  - **Methods**
    - **getInfo –** basic information about the account(aliases configured, allowed destinations, quotas)
      - Parameters : none
      - Request body example:

```
{
        "jsonrpc": "2.0",
        "id": "1",
        "method": "getInfo"
}
```

      - Response example:

```
{
    "jsonrpc": "2.0",
    "id": "1",
    "result": {
      "username": "demo",
      "enabled": true,
      "aliases": [
        "TEST",
        "TEST2"
        ],
      "allowedDestinations": [
        "NATIONAL",
        "INTERNATIONAL"
        ],
      "quotas": [
        {
          "destinationType": "INTERNATIONAL",
          "tempQuota": 0,
          "defaultQuota": 1000
        },
        {
          "destinationType": "NATIONAL",
          "tempQuota": 123,
          "defaultQuota": 10000
        }
        ]
    }
}
```

- # Sms
  - ## URL : https://smsapi.rcs-rds.ro/jsonrpc/public/sms/v2
  - ## Methods
    - **sendSms** - method used for sending a message

      Parameters:

      smsRequest object which contains:

      originator(String) – the source of the message (it can be only one of the aliases configured on your account,either numerical or alpha-numeric)

      recipient(String) – the destination of the message(the correct format of the recipient number is INTERNATIONAL, for example : 40770123456

      text(String) – the text message

      Optional parameters:

      dcs(Integer) – Data Coding Scheme (can be either 0 – default GSM7BIT or 8 - UCS-2)

      priority(Integer) – Priority (0-3), 0 lowest , 3 highest

      requestDeliveryReport(boolean) – if a statusreport should be sent via the callback URL (default is false)

      For concatenated messages only :

      partNo(Integer) – the part number

      totalParts(Integer) - the total number of parts that compose the concatenated message

      partRef(Integer0-255) - reference which should be the same for all the parts of a concatenated message

      Request example:

```
{
"jsonrpc": "2.0",
"id": "1",
"method": "sendSms",
"params": {
 "smsRequest": {
   "originator": "TEST",
   "recipient": "40770123456",
   "text": "mesaj de test"
}
}
}
```

      Response example:

```
{
   "jsonrpc": "2.0",
   "id": "1",
   "result": {
      "messageId": "6e18c9bf",
      "operator": "RCS & RDS",
      "errorCode": "0",
      "errorMessage":""
   }
}
```

      The response will contain:

      messageId – the id assigned to this message by the server. You should save them as they are needed for getStatus/cancelSms operations, to correctly identify this message.

      operator(String) – the network operator of the destination number.

      Example of an error response:

```
{
   "jsonrpc": "2.0",
   "id": "1",
   "error": {
      "code": -32002,
      "message": "Invalid alias"
   }
}
```

- **sendSmsExtended** - alternative to the sendSms method. The difference is that you can send long messages and they will get automatically splitted by the server. The request is identical, and in the response you will receive a single messageId(which identifies the whole concatenated message – if it was splitted in multiple parts).

Parameters:
smsRequest object which contains:
originator(String) – the source of the message (it can be only one of the aliases configured on your account,either numerical or alpha-numeric)
recipient(String) – the destination of the message(the correct format of the recipient number is INTERNATIONAL, for example : 40770123456
text(String) – the text message

Optional parameters:
dcs(Integer) – Data Coding Scheme (can be either 0 – default GSM7BIT or 8 - UCS-2)
priority(Integer) – Priority (0-3), 0 lowest , 3 highest
requestDeliveryReport(boolean) – if a statusreport should be sent via the callback URL (default is false)

Request example:

```
{
"jsonrpc": "2.0",
"id": "1",
"method": "sendSmsExtended",
"params": {
 "smsRequest": {
   "originator": "TEST",
   "recipient": "40770123456",
   "text": "text message longer than 160characters[...]"
}
}
}
```

Response example:

```
{
   "jsonrpc": "2.0",
   "id": "1",
   "result":
   {
      "messageId": "6e18c9bf",
      "operator": "RCS & RDS",
      "errorCode": "0",
      "errorMessage":""
   }
}
```

* Example of a long message that was split in 2 parts.

- **getStatus** – method used for checking the status of one or more messages. You can call this method for a maximum of 10 messages in bulk.

Parameters:
a list of messageId's received in the response of sendSms operation. The messageId uniquely identifies a message.

Request example:

```
{
  "jsonrpc": "2.0",
  "id": "1",
  "method": "getStatus",
  "params": {
  "statusRequest":[
  {
    "messageId": "63e1d043"
  },
  {
    "messageId": "504920ef"
  }
  ]}
}
```

Response example:

```
{
   "jsonrpc": "2.0",
   "id": "1",
   "result": [
      {
         "messageId": "63e1d043",
         "status": "U",
         "submitDate": null,
         "doneDate": null,
```

```
      "errorCode":0,
      "errorMessage":""
    },
    {
      "messageId": "504920ef",
      "status": null,
      "submitDate": null,
      "doneDate": null,
      "errorCode":0,
      "errorMessage":""
    }
  ]
}
```


The response contains the current status of the message

(U)NSENT – intial status, new message.
(Q)UEUED – in queue, en route to SMSC
(S)ENT – sucessfully submitted to SMSC
(F)AILED – cannot be submitted to SMSC(rejected)
(A)BORTED – negative delivery report received from SMSC
(D)ELIVERED – succesfull delivery report received from SMSC

submitDate - represents the datetime when the message was sucesfully submitted to the SMSC
doneDate - represents the datetime when the delivery report was received from the SMSC

Both dates are formatted identical with the SMPP Delivery Receipt standard : yyMMDDHHmm

errorCode – 0 for success, anything else should be considered an error.
For example, for an invalid messageId, you will receive :


```
{
  "jsonrpc": "2.0",
  "id": "1",
  "result": [
    {
      "messageId": "63e1d043",
      "status": null,
      "submitDate": null,
      "doneDate": null,
      "errorCode":9,
      "errorMessage":"Sms not found"
    }
  ]
}
```


- **cancelSms –** method used for canceling one or more messages. You can call this method for a maximum of 10 messages in bulk. Not all messages can be cancelled, for instance if the message has already been submitted to the SMSC.

Parameters: Same parameters as the getStatus operation : a list of messageId's received in the response of sendSms operation. The messageId uniquely identifies a message.

Request example:


```
{
  "jsonrpc": "2.0",
  "id": "1",
  "method": "cancelSms",
  "params": {
    "cancelRequest": [
      {
        "messageId": "154b0d56"
      },
      {
        "messageId": "63e1d043",
      }
        ]
  }

}
```


Response example:


```
{
  "jsonrpc": "2.0",
```

```
  "id": "1",
  "result": [
     {
        "errorCode": 9,
        "errorMessage": "Sms not found",
        "messageId": "154b0d56"
     },
     {
        "errorCode": 9,
        "errorMessage": "Sms not found",
        "messageId": "63e1d043"
     }
  ]
}
```

- **Callback** for receiving status reports

Your account can have one callback http url associated.
For all the messages submitted via **sendSms**/**sendSmsExtended** with the flag **requestDeliveryReport** set to "**true**", the system will send a HTTP POST request to your URL in JSON-RPC format. Your server should respond with a HTTP status code 200(OK) to acknowledge receiving the status report.

The body of the request is almost identical to a response for **getStatus** method :

```
{
  "jsonrpc": "2.0",
  "id": "1",
  "result":   {
     "messageId": c1040ba6ea99,
     "status": "A",
     "submitDate": "2006051624",
     "doneDate": "2006051624"
  }
}
```

You can still call the **getStatus** method to interrogate the status of an sms on-demand.

RESPONSE API CODES:

```
SUCCESS(0),
INVALID_SYSTEMID(1),
INVALID_ALIAS(2),
INVALID_DCS(3),
INVALID_TEXT(4),
INVALID_RECIPIENT(5),
INVALID_RECIPIENT_IS_BLACKLISTED(6),
INVALID_RECIPIENT_NOT_ALLOWED(7),
MAX_QUOTA_REACHED(8),
SMS_NOT_FOUND(9),
SMS_NOT_CANCELABLE(10),
TOO_MANY_PARAMETERS(11),
SYSTEM_ERROR(12),
USER_NOT_FOUND(13),
INVALID_PRIORITY(14)
```

- In case of non-bulk operations, api response codes will be mapped to json-rpc custom error codes :

```
INVALID_SYSTEMID(-32001),
INVALID_ALIAS(-32002),
INVALID_DCS(-32003),
INVALID_TEXT(-32004),
INVALID_RECIPIENT(-32005),
INVALID_RECIPIENT_IS_BLACKLISTED(-32006),
INVALID_RECIPIENT_NOT_ALLOWED(-32007),
MAX_QUOTA_REACHED(-32008),
SMS_NOT_FOUND(-32009),
SMS_NOT_CANCELABLE(-32010),
TOO_MANY_PARAMETERS(-32011),
SYSTEM_ERROR(-32012),
USER_NOT_FOUND(-32013),
INVALID_PRIORITY(-32014)
AUTH_FAILURE(-32097),
ACCESS_DENIED(-32098),
TOO_MANY_REQUESTS(-32099),
INTERNAL_ERROR(-32603)
```