REMOTE365

Web Service API

Version 1.13

link mobility

Table of Contents

REMOTE365

This document describes the web service that can be used to access REMOTE365 functionality. With REMOTE365, MT tests are made available to any kind of 3rd party client.

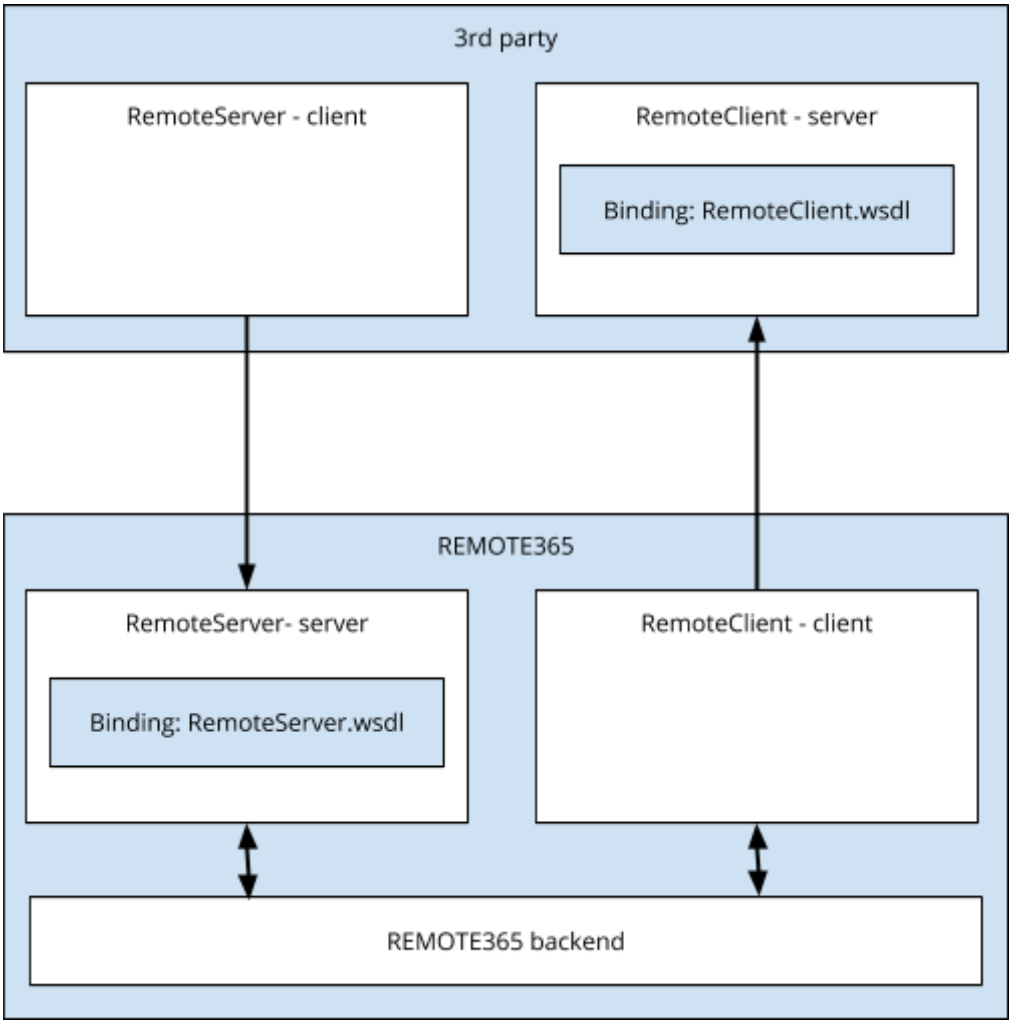
The REMOTE365 Web Service is available at https:/remote365.net:17365/RemoteServerPort

and is secured through SSL encryption and HTTP basic authentication. To use the web service, the 3rd party client must have a valid account at REMOTE365. A standard account is created by signing up at the website https://remote365.net or by joining the professional customer program.

The HTTP basic authentication data needs to be provided each time the client accesses any of the web service's server methods.

Each IP has a maximum number of requests per time frame that can be sent to the server. If this amount is exceeded, the client will not be able to send any request to the server within a specific time frame.

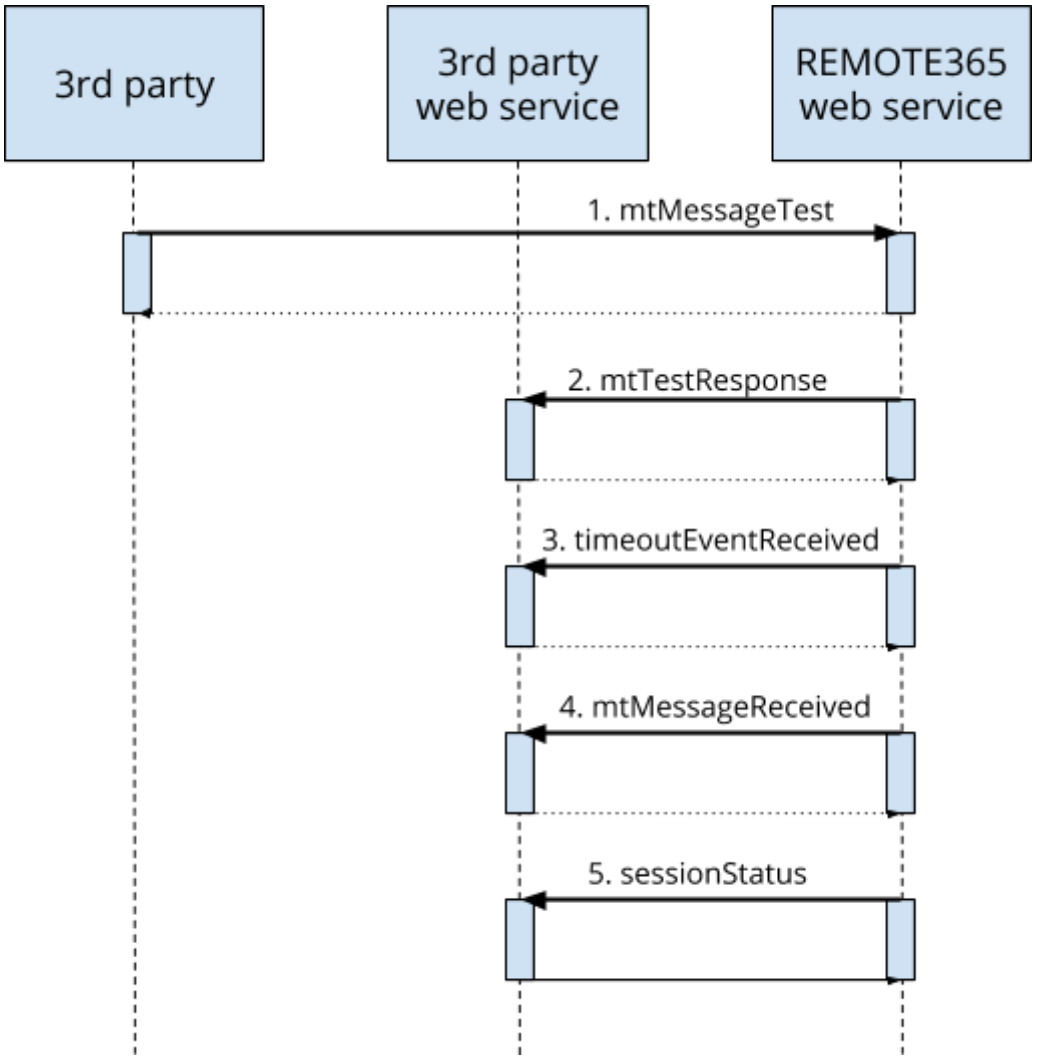
Component overview



MT Test

This chapter describes how to use the web service API to perform MT tests. It also contains information on the requests that need to be sent to the web service as well as requests that are sent to the 3rd party web service.

Reading this chapter requires that the 3rd party has read carefully and understood the WSDL file (both, for server and client). The 3rd party should have some technical experience with web services, as the client also needs to implement a web service server himself for callback requests of REMOTE365.



1. The 3rd party sends the request “mtMessageTest” to the REMOTE365 web service. REMOTE365 checks all parameters and returns a reference id for later callbacks to the 3rd party client. If an error occurs, an exception will be generated with a specific error code. If no exception is raised, the process will start within the backend.
2. REMOTE365 calls the 3rd party’s web service with the method “mtTestResponse”. The reference id generated in step (1) is transmitted and an errorcode. If the error code is zero, the transmitted session id and MSISDN should be used to start sending a message for testing purposes.
    a. After receiving the session id, the 3rd party should send the SMS immediately to the given MSISDN (it needs to be made sure that the session id is inserted into the message body).
3. If one to three test cases were specified and no SMS was received during the given time frame REMOTE365 will call the 3rd party’s web service with timeoutEventReceived method.
4. After receiving the SMS, REMOTE365 calls the 3rd party’s web service with the method “mtMessageReceived”. All message parameters are transmitted to the 3rd party.

5. The "sessionStatus" method is called by REMOTE365 at the 3rd party's web service in case an MT Test is not available anymore. Therefore, the reference id is transmitted to enable the 3rd party to remove any MT Test specific data. After "sessionStatus" is called at 3rd party, no more requests will be sent to the 3rd party for the given reference id (session is closed).

Protocol description

The following chapter describes all server and client methods.

REMOTE365 server protocol

All method calls to REMOTE365 web service are HTTPS secured. The 3rd party needs to set HTTP basic authentication header information for each request. The following chapter describes all server methods that are defined in RemoteServer.wsdl.

mtMessageTest method

This method is invoked by 3rd party to initiate an MT test. The method returns: MtMessageAnswer. It contains the referenceId for the MT test session. All following client callbacks contain the generated reference Id. The reference id is unique in the system. Please note that a session for an MT test is not useable before the mtTestResponse callback to the client has been issued.
Further it is possible to force REMOTE365 to create timeout events in case an SMS did not arrive in a specific time span. Therefore you can define up to three testcases within the mtMessageTest method call. For each testcase REMOTE365 will create an own timeout event. You can refer a testcase by a reference id of your own choice. The timeout event contains the duration in seconds the modem was offline in the given time span. This way you can evaluate if the missing SMS might be the result of the modem not being online for a certain time and not due to delivery problems of your SMS supplier.[1]

If the mtMessageTest call is not successful, the following exception may be raised:

- RemoteServiceException, containing an error code and an (optional) error description

The following table shows the parameter structure of the mtMessageTest method:

Parameter Name	SubParam Name	Type	Presence	Description
simNwc		int	Mandatory	Target network code. Must be valid. You can get the valid network codes by calling the getAvailableNwc method or by browsing https://status.remote365.net
externalReference		String	Mandatory	Max length: 50 chars. Contains an external reference from 3rd party. This reference id is transmitted with each request to the 3rd party web service client. It is possible to set the parameter as an empty string.
messageType		int	Mandatory	Type of message for MT test that is used later for testing. Currently only SMS is available: *1 – SMS Message*

[1] GSM devices that receive hundreds of messages every day for testing purposes might get blocked by the operator due to fraud prevention. Since local operators have different ways of communicating with their registered devices, which might lead to the modem occasionally not having a network connection or requiring a restart, we offer you this option to re-evaluate the quality of your tested route.

testcases		Testcases	Optional	It is possible to define zero to three testcases. A Testcase will force REMOTE365 to create a timeout event if no SMS arrives to a given point in time.
	item	Testcase Item	Optional Max.3 items	A single testcase.
	key (sub param of item)	String	Mandatory	Reference id which has to be unique per request.
	value (sub param of item)	int	Mandatory	Denotes the point in time relative to the current time in seconds.

Possible error codes

Calling the method may result in an exception thrown by the server. The thrown exception is of type RemoteServiceException. It contains an error code and an (optional) error description. Following error codes may occur:

Error code	Description
1	No connection to backend systems. For further information see error description.
2	Authentication failed (wrong or missing login credentials in HTTP basic authentication header or no callback URL configured).
3	Error while requesting the MSISDN for MT test (the process responsible for creating the process raises errors or the NWC is faulty).
4	Backend error (database error e.g.).
6	Wrong or missing parameter (network code is zero or external reference is too long or more than three testcase items).
7	Not enough credit. The test cannot be started due to insufficient credit in prepaid case.
8	Network code not valid. See https://status.remote365.net for all valid NWCs.
100	Exceptional error in backend (technical support should be contacted).

Example request

Following request is a valid mtMessageTest request:

```
POST /RemoteServerPort HTTP/1.0
Content-Type: text/xml;charset=UTF-8
SOAPAction: ""
Authorization: Basic dGVzdDpuZXVlc3Bhc3N3b3J0
User-Agent: Jakarta Commons-HttpClient/3.1
Host: remote365.net:17365
Content-Length: 380

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:rem="rem">
  <soapenv:Header/>
  <soapenv:Body>
    <rem:mtMessageTest>
      <rem:simNwc>26202</rem:simNwc>
      <rem:externalReference>myRef</rem:externalReference>
      <rem:messageType>1</rem:messageType>
      <rem:testcases>
        <x-:item>
          <x-:key>myTestcaseRefId1</x:-key>
          <x-:value>360</x-:value>
        </x-:item>
        <x-:item>
          <x-:key>myTestcaseRefId2</x:-key>
```

```
            <x-:value>720</x-:value>
          </x-:item>
        </rem:testcases>
      </rem:mtMessageTest>
    </soapenv:Body>
</soapenv:Envelope>
```

Example successful response

Following response is received by client in case of successful request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <mtMessageTestResponse xmlns="rem">
      <mtMessageTestReturn>
        <referenceId>122884427fb7d3712415a72410</referenceId>
      </mtMessageTestReturn>
    </mtMessageTestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Example error response

Following response is received in case of an erroneous request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server.generalException</faultcode>
      <faultstring/>
      <detail>
        <ns1:fault xmlns:ns1="rem">
          <ns1:errorCode>8</ns1:errorCode>
          <ns1:errorDescription>NWC not allowed.</ns1:errorDescription>
        </ns1:fault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

getAvailableNwcs method

This method is invoked by 3rd party to get all allowed network codes for MT tests. It is proposed to call this method in regular intervals, to ensure having an actual amount of network codes available.

All network codes can also be retrieved manually at https://status.remote365.net.

If the mtMessageTest call is unsuccessful, the following exception may be raised:

- RemoteServiceException, containing an error code and an (optional) error description (see possible error codes)

It returns an array of network codes that can be used to identify operator and according network code to be used in MT tests.

This method is secured (as all other web service methods) through HTTP basic authentication. This means the 3rd party also needs to transmit the login and password in the HTTP Header Authentication fields!

The following table shows the return parameter structure of the getAvailableNwcs method:

Parameter Name	SubParam Name	Type	Presence	Description
ArrayOfNwcType (array)	countryIso	String	Mandatory	The 3-digit ISO 3166-1 alpha-3 country code.
	nwc	int	Mandatory	The network code itself.
	operatorName	String	Mandatory	The name of the operator.

Possible error codes

Calling the method may result in an exception thrown by the server. The thrown exception is of type RemoteServiceException. It contains an error code and an (optional) error description. Following error codes may occur:

Error code	Description
1	No connection to backend systems. For further information see error description.
2	Authentication failed (wrong or missing login credentials in HTTP basic authentication header).
100	Exceptional error in backend (technical support should be contacted).

Example request

Following request is a valid getAvailableNwcs request:

```
POST /RemoteServerPort HTTP/1.0
Content-Type: text/xml;charset=UTF-8
SOAPAction: ""
Authorization: Basic dGVzdDpuZXVlc3Bhc3N3b3J0
User-Agent: Jakarta Commons-HttpClient/3.1
Host: www.remote365.net:17365
Content-Length: 205

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:rem="rem">
  <soapenv:Header/>
  <soapenv:Body>
    <rem:getAvailableNwcs/>
  </soapenv:Body>
</soapenv:Envelope>
```

Example response

Following response is received by client in case of successful request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getAvailableNwcsResponse xmlns="rem">
      <getAvailableNwcsReturn>
        <countryIso>CHE</countryIso>
        <nwc>22801</nwc>
        <operatorName>Swisscom Ltd</operatorName>
      </getAvailableNwcsReturn>
      <getAvailableNwcsReturn>
        <countryIso>AUT</countryIso>
        <nwc>23203</nwc>
```

```
        <operatorName>T-Mobile Austria</operatorName>
      </getAvailableNwcsReturn>
      <getAvailableNwcsReturn>
        <countryIso>DEU</countryIso>
        <nwc>26201</nwc>
        <operatorName>T-Mobile Deutschland</operatorName>
      </getAvailableNwcsReturn>
    </getAvailableNwcsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Example error response

Following response is received in case of an erroneous request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>soapenv:Server.generalException</faultcode>
      <faultstring/>
      <detail>
        <faultData>
          <ns1:errorCode xmlns:ns1="rem">2</ns1:errorCode>
          <ns2:errorDescription xmlns:ns2="rem">Authentication failure.</ns2:errorDescription>
        </faultData>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

REMOTE365 client protocol

This client is invoked by the REMOTE365 for results on MT test processes. The 3rd party needs to implement a web service server with RemoteClient.wsdl. It is recommended to implement these services HTTPS secured. The REMOTE365 server will not check any server certificates in case of HTTPS communication.

Configuring the callback URL

The callback URL can be configured on the website. Navigate to Configuration / API Connection and enter a valid URL. HTTPS is highly recommened. Please make sure to specify the port if it differs from 80, for instance when using HTTPS (port 443).

mtTestResponse method

This callback method is invoked by the REMOTE365 server if a session for MT test is established within the backend. After calling this method successfully (and of course if a session has been established), billing is initiated by REMOTE365 at the 3rd party account. It is highly recommended that this method never raises any exceptions (such as RemoteException).

REMOTE365 will not use any HTTP basic authentication for this request. If the response HTTP status code is **not** 200 OK or the response is an exception, the REMOTE365 server will recall this method ten times in an interval of ten minutes and finally discard it, if retrying wasn't successful.

The following table shows the parameter structure of the mtTestResponse method:

Parameter Name	Type	Presence	Description
msisdn	String	Mandatory	Target MSISDN the SMS should be sent to.
sessionId	String	Mandatory	This session id has been generated by REMOTE365. It needs to be inserted at the beginning of the message body of the SMS (see appendix 5.3).
externalReference	String	Mandatory	External reference provided with mtMessageTest request to REMOTE365 before.
referenceId	String	Mandatory	Reference id generated by REMOTE365 after successful call of mtMessageTest.
errorCode	int	Mandatory	See possible error codes.

Possible error codes

Error code	Description
0	MT test session established. 3rd party should sent MT test message.
4	Backend error (database error e.g.).
5	No MSISDN found for provided NWC (no modem available for the given NWC at this time.). See https://status.remote365.net for a network status overview.
100	Exceptional error occurred in backend (technical support should be contacted).

Example request

Following request is a valid mtTestResponse request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
```

```
   <mtTestResponse xmlns="rem">
    <msisdn>00491702007810</msisdn>
    <sessionId>1248435245080</sessionId>
    <externalReference>myRef</externalReference>
    <referenceId>122884427fb7d3712415a72410</referenceId>
    <errorCode>0</errorCode>
   </mtTestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

timeoutEventReceived method

This callback method is invoked by the REMOTE365 server if a timeout event was created after a SMS did not arrive in time. The corresponding external reference and reference id for the test session is transmitted.

REMOTE365 will not use any HTTP basic authentication for this request. If the response HTTP status code is **not** 200 OK or the response is an exception, the REMOTE365 server will recall this method ten times in an interval of ten minutes and finally discard it, if retrying wasn't successful.

The following table shows the parameter structure of the timeouteventResponse method:

Parameter Name	Type	Presence	Description
msisdn	String	Mandatory	Target MSISDN the SMS was sent to.
sessionId	String	Mandatory	The session id for the test this timeout was created for.
referenceId	String	Mandatory	Reference id returned in the response on the mtMessageTest Request.
runtimeRefId	String	Mandatory	Customer reference id for this timeout event.
offDuration	int	Mandatory	Duration in seconds the modem was offline.

Example request

Following request is a valid timeoutEventReceived request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <timeoutEventReceived xmlns="rem">
      <timeout>
        <externalReference>myReferenceId</externalReference>
        <referenceId>1369c4626991899c2421d80ebf8</referenceId>
        <runtimeRefId>myRuntimeReferenceId</runtimeRefId>
        <sessionId>12569</sessionId>
        <msisdn>+4915258765506</msisdn>
        <offDuration>15</offDuration>
      </timeout>
    </timeoutEventReceived>
  </soapenv:Body>
</soapenv:Envelope>
```

mtMessageReceived method

This callback method is invoked by the REMOTE365 server if an MT message has been received. The corresponding external reference and reference id for the test session is transmitted. All parameters for the received message are transmitted.

REMOTE365 will not use any HTTP basic authentication for this request. If the response HTTP status code is **not** 200 OK or the response is an exception, the REMOTE365 server will recall this method ten times in an interval of ten minutes and finally discard it, if retrying wasn't successful.

The following table shows the parameter structure of the messageReceived method:

Parameter Name	SubParam Name	Type	Presence	Description
referenceId		String	Mandatory	Reference id for test session.
externalReference		String	Mandatory	External reference given by initiating the test session with the call of mtMessageTest at REMOTE365 probe.
msgBody		BodyType	Mandatory	Contains all message body relevant data.
	charset	String	Optional	Not used yet.
	userdata	String	Mandatory	Contains the message body itself.
SIMnwc		int	Mandatory	The network code the MSISDN is located at.
receiveTime		DateTime	Mandatory	The timestamp the message was received at REMOTE365 probe.
originator		String	Mandatory	Originator of the message..
originatorTON		String	Mandatory	Originator's type of number (see appendix 5.4.3).
dcs		int	Optional	Data-Coding-Scheme for binary SMS.
udh		String	Optional	User data header for binary (e.g. WAP Push) messages.
smsc		String	Mandatory	Short Message Service Center. This is the SMSC the message was delivered through.
scts		DateTime	Mandatory	This field contains the date when the message was sent or, if not available, submitted from the operator to REMOTE365 probe.
receiveCount		int	Mandatory	In case that multiple messages have been received for a specific session, the current receive count is transmitted.
sessionId		String	Mandatory	Defines the test session the message is associated with.
pdusAsString		String	Optional	Protocol Data Unit(s) separated by ";" (in case of concat SMS).

Example request

Following request is a valid mtMessageRecieved request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
```

```
  <mtMessageReceived xmlns="rem">
   <referenceId>133efe41932d57031bbbfe80adc</referenceId>
   <externalReference>ExtId</externalReference>
   <msgBody>
    <charset xsi:nil="true"/>
    <encoding xsi:nil="true"/>
    <userdata>11042 Test answer</userdata>
   </msgBody>
   <simNwc>26201</simNwc>
   <receiveTime xsi:type="xsd:dateTime">2011-11-29T15:15:45.000Z</receiveTime>
   <originator>Test</originator>
   <originatorTON>1</originatorTON>
   <dcs>0</dcs>
   <udh></udh>
   <smsc xsi:nil="true"/>
   <scts xsi:nil="true"/>
   <receiveCount>1</receiveCount>
   <sessionId>11042</sessionId>
   <pdusAsString>
0791947101673054040D91945111663746F2000011119261519340 11B1188C260351CB733A28EC9EDFCB
72
   </pdusAsString>
  </mtMessageReceived>
 </soapenv:Body>
</soapenv:Envelope>
```

sessionStatus method

This method is invoked by REMOTE365 server if a test session has been changed. This method is only invoked if a test session is closed at REMOTE365. This means there will be no further method calls for the transmitted reference id by REMOTE365. Session can be released at client side. The corresponding external reference and reference id for the test session is transmitted.

REMOTE365 will not use any HTTP basic authentication for this request. If the response HTTP status code is **not** 200 OK or the response is an exception, the REMOTE365 server will recall this method ten times in an interval of ten minutes and finally discard it, if retrying wasn't successful.

The following table shows the parameter structure of the sessionStatus method:

Parameter Name	SubParam Name	Type	Presence	Description
status		SessionStatus	Mandatory	Contains information on the session status method call.
	statusCode	int	Mandatory	The session change status code (only one implemented currently): 1 – session closed
	statusDescription	String	Mandatory	Description of the sessionStatus method call.
	referenceId	String	Mandatory	The reference id for the session the method is called for.
	externalReference	String	Mandatory	Contains the external reference given by initiated test call at REMOTE365 by 3rd party.

Example request

Following request is a valid sessionStatus request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <sessionStatus xmlns="rem">
   <status>
    <statusCode>1</statusCode>
    <statusDescription>Session closed. Reference inactive now.</statusDescription>
    <referenceId>12287c48ac2e6157be17fe4bad6</referenceId>
    <externalReference>myExtReference</externalReference>
   </status>
  </sessionStatus>
 </soapenv:Body>
</soapenv:Envelope>
```

Appendix

REMOTE365 server WSDL

The REMOTE365 Server WSDL can be downloaded from server location at

https://remote365.net:17365/RemoteServerPort?wsdl

The WSDL can also be copied from here:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="rem" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="rem"
xmlns:intf="rem" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
 <schema elementFormDefault="qualified" targetNamespace="rem" xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://xml.apache.org/xml-soap"/>
  <element name="mtMessageTest">
   <complexType>
    <sequence>
     <element name="simNwc" type="xsd:int"/>
     <element name="externalReference" type="xsd:string" minOccurs="0"/>
     <element name="messageType" type="xsd:int"/>
     <element name="testcases" type="apachesoap:Map" minOccurs="0"/>
    </sequence>
   </complexType>
  </element>
  <element name="mtMessageTestResponse">
   <complexType>
    <sequence>
     <element name="mtMessageTestReturn" type="impl:MtMessageAnswer"/>
    </sequence>
   </complexType>
  </element>
  <complexType name="MtMessageAnswer">
   <sequence>
    <element name="referenceId" nillable="true" type="xsd:string"/>
   </sequence>
  </complexType>
  <complexType name="RemoteServiceException">
   <sequence>
    <element name="errorCode" type="xsd:int"/>
    <element name="errorDescription" nillable="true" type="xsd:string"/>
   </sequence>
  </complexType>
  <element name="fault" type="impl:RemoteServiceException"/>
  <element name="getAvailableNwcs">
   <complexType/>
  </element>
  <element name="getAvailableNwcsResponse">
```

```
  <complexType>
   <sequence>
    <element maxOccurs="unbounded" name="getAvailableNwcsReturn" type="impl:NwcType"/>
   </sequence>
  </complexType>
 </element>
 <complexType name="NwcType">
  <sequence>
   <element name="countryIso" nillable="true" type="xsd:string"/>
   <element name="nwc" type="xsd:int"/>
   <element name="operatorName" nillable="true" type="xsd:string"/>
  </sequence>
 </complexType>
</schema>
<schema elementFormDefault="qualified" targetNamespace="http://xml.apache.org/xml-soap"
xmlns="http://www.w3.org/2001/XMLSchema">
 <import namespace="rem"/>
 <complexType name="mapItem">
  <sequence>
   <element name="key" nillable="true" type="xsd:anyType"/>
   <element name="value" nillable="true" type="xsd:anyType"/>
  </sequence>
 </complexType>
 <complexType name="Map">
  <sequence>
   <element maxOccurs="3" minOccurs="0" name="item" type="apachesoap:mapItem"/>
  </sequence>
 </complexType>
</schema>
</wsdl:types>
 <wsdl:message name="RemoteServiceException">
    <wsdl:part element="impl:fault" name="fault"/>
 </wsdl:message>
 <wsdl:message name="mtMessageTestRequest">
    <wsdl:part element="impl:mtMessageTest" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getAvailableNwcsResponse">
    <wsdl:part element="impl:getAvailableNwcsResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="mtMessageTestResponse">
    <wsdl:part element="impl:mtMessageTestResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="getAvailableNwcsRequest">
    <wsdl:part element="impl:getAvailableNwcs" name="parameters"/>
 </wsdl:message>
 <wsdl:portType name="RemoteServer">
    <wsdl:operation name="mtMessageTest">
       <wsdl:input message="impl:mtMessageTestRequest" name="mtMessageTestRequest"/>
       <wsdl:output message="impl:mtMessageTestResponse" name="mtMessageTestResponse"/>
       <wsdl:fault message="impl:RemoteServiceException" name="RemoteServiceException"/>
    </wsdl:operation>
    <wsdl:operation name="getAvailableNwcs">
       <wsdl:input message="impl:getAvailableNwcsRequest" name="getAvailableNwcsRequest"/>
       <wsdl:output message="impl:getAvailableNwcsResponse" name="getAvailableNwcsResponse"/>
    </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="RemoteServerPortSoapBinding" type="impl:RemoteServer">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="mtMessageTest">
       <wsdlsoap:operation soapAction=""/>
       <wsdl:input name="mtMessageTestRequest">
          <wsdlsoap:body use="literal"/>
       </wsdl:input>
       <wsdl:output name="mtMessageTestResponse">
          <wsdlsoap:body use="literal"/>
       </wsdl:output>
       <wsdl:fault name="RemoteServiceException">
          <wsdlsoap:fault name="RemoteServiceException" use="literal"/>
       </wsdl:fault>
    </wsdl:operation>
```

```
    <wsdl:operation name="getAvailableNwcs">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="getAvailableNwcsRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getAvailableNwcsResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RemoteServerService">
    <wsdl:port binding="impl:RemoteServerPortSoapBinding" name="RemoteServerPort">
      <wsdlsoap:address location="https://www.remote365.net:17365/RemoteServerPort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

REMOTE365 client WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="rem" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="rem"
xmlns:intf="rem" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
 <schema elementFormDefault="qualified" targetNamespace="rem" xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="mtMessageReceived">
   <complexType>
    <sequence>
     <element name="referenceId" type="xsd:string"/>
     <element name="externalReference" type="xsd:string" minOccurs="0"/>
     <element name="msgBody" type="impl:BodyType"/>
     <element name="simNwc" type="xsd:int"/>
     <element name="receiveTime" type="xsd:dateTime"/>
     <element name="originator" type="xsd:string"/>
     <element name="originatorTON" type="xsd:int"/>
     <element name="dcs" type="xsd:int" minOccurs="0"/>
     <element name="udh" type="xsd:string" minOccurs="0"/>
     <element name="smsc" type="xsd:string"/>
     <element name="scts" type="xsd:dateTime"/>
     <element name="receiveCount" type="xsd:int"/>
     <element name="sessionId" type="xsd:string"/>
     <element name="pdusAsString" type="xsd:string" minOccurs="0"/>
    </sequence>
   </complexType>
  </element>
  <complexType name="BodyType">
   <sequence>
    <element name="charset" nillable="true" type="xsd:string"/>
    <element name="encoding" nillable="true" type="xsd:string" minOccurs="0"/>
    <element name="userdata" nillable="true" type="xsd:string" minOccurs="0"/>
   </sequence>
  </complexType>
  <element name="mtMessageReceivedResponse">
   <complexType/>
  </element>
  <element name="mtTestResponse">
   <complexType>
    <sequence>
     <element name="msisdn" type="xsd:string"/>
     <element name="sessionId" type="xsd:string"/>
     <element name="externalReference" type="xsd:string" minOccurs="0"/>
     <element name="referenceId" type="xsd:string"/>
     <element name="errorCode" type="xsd:int"/>
    </sequence>
```

```
 </complexType>
</element>
<element name="mtTestResponseResponse">
 <complexType/>
</element>
<element name="sessionStatus">
 <complexType>
  <sequence>
   <element name="status" type="impl:SessionStatus"/>
  </sequence>
 </complexType>
</element>
<complexType name="SessionStatus">
 <sequence>
  <element name="statusCode" type="xsd:int"/>
  <element name="statusDescription" nillable="true" type="xsd:string"/>
  <element name="referenceId" nillable="true" type="xsd:string"/>
  <element name="externalReference" nillable="true" type="xsd:string"/>
 </sequence>
</complexType>
<element name="sessionStatusResponse">
 <complexType/>
</element>
<element name="timeoutEventReceived">
 <complexType>
  <sequence>
   <element name="timeout" type="impl:TimeoutEvent"/>
  </sequence>
 </complexType>
</element>
<complexType name="TimeoutEvent">
 <sequence>
  <element name="externalReference" nillable="true" type="xsd:string"/>
  <element name="referenceId" nillable="true" type="xsd:string"/>
  <element name="runtimeRefId" nillable="true" type="xsd:string"/>
  <element name="sessionId" nillable="true" type="xsd:string"/>
  <element name="msisdn" nillable="true" type="xsd:string"/>
  <element name="offDuration" type="xsd:int"/>
 </sequence>
</complexType>
<element name="timeoutEventReceivedResponse">
 <complexType/>
</element>
</schema>
</wsdl:types>
 <wsdl:message name="timeoutEventReceivedResponse">
    <wsdl:part element="impl:timeoutEventReceivedResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="mtMessageReceivedResponse">
    <wsdl:part element="impl:mtMessageReceivedResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="timeoutEventReceivedRequest">
    <wsdl:part element="impl:timeoutEventReceived" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="sessionStatusResponse">
    <wsdl:part element="impl:sessionStatusResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="sessionStatusRequest">
    <wsdl:part element="impl:sessionStatus" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="mtTestResponseResponse">
    <wsdl:part element="impl:mtTestResponseResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="mtTestResponseRequest">
    <wsdl:part element="impl:mtTestResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="mtMessageReceivedRequest">
    <wsdl:part element="impl:mtMessageReceived" name="parameters"/>
 </wsdl:message>
 <wsdl:portType name="RemoteClient">
```

```
    <wsdl:operation name="mtMessageReceived">
      <wsdl:input message="impl:mtMessageReceivedRequest" name="mtMessageReceivedRequest"/>
      <wsdl:output message="impl:mtMessageReceivedResponse" name="mtMessageReceivedResponse"/>
    </wsdl:operation>
    <wsdl:operation name="mtTestResponse">
      <wsdl:input message="impl:mtTestResponseRequest" name="mtTestResponseRequest"/>
      <wsdl:output message="impl:mtTestResponseResponse" name="mtTestResponseResponse"/>
    </wsdl:operation>
    <wsdl:operation name="sessionStatus">
      <wsdl:input message="impl:sessionStatusRequest" name="sessionStatusRequest"/>
      <wsdl:output message="impl:sessionStatusResponse" name="sessionStatusResponse"/>
    </wsdl:operation>
    <wsdl:operation name="timeoutEventReceived">
      <wsdl:input message="impl:timeoutEventReceivedRequest" name="timeoutEventReceivedRequest"/>
      <wsdl:output message="impl:timeoutEventReceivedResponse" name="timeoutEventReceivedResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="RemoteClientPortSoapBinding" type="impl:RemoteClient">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="mtMessageReceived">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="mtMessageReceivedRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="mtMessageReceivedResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="mtTestResponse">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="mtTestResponseRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="mtTestResponseResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="sessionStatus">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="sessionStatusRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="sessionStatusResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="timeoutEventReceived">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="timeoutEventReceivedRequest">
        <wsdlsoap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="timeoutEventReceivedResponse">
        <wsdlsoap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RemoteClientService">
    <wsdl:port binding="impl:RemoteClientPortSoapBinding" name="RemoteClientPort">
      <wsdlsoap:address location="https://anyurl/RemoteClientPort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

How to insert the session id for MT Test SMS

REMOTE365 generates a unique session id. This session id and a destination MSISDN will sent by mtTestResponse method after a MT Test has been requested. The session id is only valid for eight days in combination with the corresponding MSISDN.

To perform a test with REMOTE365 the SMS MT sent by you must contain the session id. SMS MT can't be assigned if the session id is not included or truncated within the message.

For text SMS the session id must be inserted as the first part of the e.g.: "12345Hello World".

For 8 bit binary SMS the session id must be encoded within the binary message e.g. (IA5): "313233343548656C6C6F20576F6C7264".

For Unicode SMS (16 bit) the session id must be encoded within the 16Bit message e.g. (ia5): "00310032003300340035004800650065006C006C006F00200057006F006C00720064".

Glossary

3rd Party

User of the web service

HTTP basic authentication

HTTP header information with authentication (can be provided via the HTTP protocol)

ISO country code

An up to date list of all ISO 3166-1 alpha-3 country code is maintained at Wikipedia:
https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3

MT test

Mobile Terminated test. A message is sent to a modem. The message is transmitted to the 3rd party if received.

Network code (NWC)

The network code or NWC is the combination of mobile country code (MCC) and the mobile network code (MNC).
For the German (MCC = 262) operator Telekom Deutschland (MNC = 01) the NWC is 26201. A list of all known operators can be found at Wikipedia https://en.wikipedia.org/wiki/Mobile_country_code

SMSC

Short Message Service Center – an operator platform for sending and receiving SMS

Type of number (TON)

REMOTE365 uses the following TONs:

Type of number (TON)	Description	Example
0	Unknown	-
1	International	491749443568
2	National	00491749443568 or 01749443568
3	Shortcode	86000
5	Alphanumeric	REMOTE365 (up to eleven characters)

WSDL

Web Service Description Language. A file that describes the web service structure. A web service is independent from platform and programming language.

Document history

Datum	Author	Description
13.05.2009	MIGE	First Version
29.05.2009	MIGE	Version 1.0
17.07.2009	MIGE	Version 1.1 ● Added MO Tests ● Changed existing requests and examples ● Removed network codes
28.07.2009	MIGE	Version 1.2 ● Changed Client and Server WSDL
30.07.2009	MIGE	Version 1.3 ● Added explanation how to code SessionID into MT Test SMS
15.03.2011	MIGE	Version 1.4 ● Removed MO Testing
07.06.2011	ROSC	Version 1.5 ● Changed Test ID encoding
30.11.2011	MAHA	Version 1.6 ● Added return of PDU string within MT tests
27.04.2012	ROSC	Version 1.7 ● Added testcases / timeout events
05.12.2012	JESP	Version 1.8 ● WSDL corrected
01.03.2015	HELO	Version 1.9 ● Added NWC, ISO codes and TON to appendix ● Minor updates and fixes
04.07.2016	HELO	Version 1.10 ● Added links to REMOTE365 Status Dashboard ● Clarified valid responses for callbacks to 3rd party ● Updated retry behaviour for callbacks
05.09.2016	HELO	Version 1.11 ● Updated API URL ● Updated graphics ● Minor updates and fixes
28.10.2016	HELO	Version 1.12 ● Added chapter 'Configuring the callback URL'
15.05.2017	HELO	Version 1.13 ● Adapted document to LINK Mobility Corporate Identity