TATA CONSULTANCY SERVICES
Experience certainty.

TATA

RBI NG-RTGS, Next-Generation Real Time Gross Settlement System

Annex H

September 2013, Version v1.1

Document Release Note

Notice No.:

Customer: Reserve Bank of India (RBI)

Project: Next Generation Real Time Gross Settlement System (NG-RTGS)

Document Details

Name	Version Number	Description
Annex H	**v1.0**	The document describes the details of API Web Service Interface to the Participants, the business and technical aspects associated with it.
Annex H	**V1.1**	Changes in certificate requirement

Revision Details

Action Taken (add/del/change)	Previous page number	New page number	Revision description
Changed	11	11	Change in Figure1. PO connected through INFINET network
Deleted	07	07	Alternative implementation solutions of the API interface to all participants
Addition	11	12	Purpose, Signature, Arguments, Return Value added under Support functions
Addition	12	14	Example, XSD schema and conventions added in GetcurrentTimetable() function
Addition	13	17	WSDL XML Definition and XSD Schema definition added under Web Service Parameter

Change Register serial numbers covered:

The documents or revised pages are subject to document control.

Please keep them up-to-date using the release notices from the distributor of the document.

These are confidential documents. Unauthorized access or copying is prohibited.

Approved by:	Soumyaranjan Panda	Authorised by:	Sanjay Nayak
Date:	27/02/2013	Date:	27/02/2013

Document Revision List

Customer: Reserve Bank of India (RBI)

Project: Next Generation Real Time Gross Settlement System (NG-RTGS)

Document Name: Control Specifications Document (CSD)

Release Notice Reference (for release)

Revision Number	Revision Date	Revision Description	Page Number	Previous Page Number	Action Taken	Addenda/ New Page	Release Notice Reference

About this Document

Purpose

This document describes the technical and business aspects of the Application Programming Interface (API), which needs to be known by the commercial banks so that they can implement the interface and integrate it within their existing IT environment. This Web-Service interface service will be offered to all existing Participants to allow them to exchange payment messages and acknowledgements with the NG-RTGS when the main communication over Society for Worldwide Interbank Financial Telecommunications (SWIFT) or INdian FINancial NETwork INFINET networks is not possible.

Intended Audience

This document is intended for users of NG-RTGS Reserve Bank of India (RBI).

Contents

Total number of pages in this document (including cover page) is 21.

List of Figures

List of Abbreviations

Abbreviation	Expanded Form
API	Application Programming Interface
DN	Distinguished Names
HTTPS	Secure Hyper Text Transfer Protocol
IDRBT	Institute for Development & Research in Banking Technology
IFSC	Indian Financial System Code
INFINET	INdian FINancial NETwork
NG-RTGS	Next Generation Real Time Gross Settlement
RBI	Reserve Bank of India
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
STP	Straight Through Processing
SWIFT	Society for Worldwide Interbank Financial Telecommunications

1. Introduction

The NG-RTGS application provides several independent channels of communication with the Participants for the submission and reception of the payment instructions settled in real-time. One of these channels is represented by the Web-Service interface service that will be offered to all existing Participants to allow them to exchange payment messages and acknowledgements with the NG-RTGS when the main communication over Society for Worldwide Interbank Financial Telecommunications (SWIFT) or INdian FINancial NETwork (INFINET) networks is not possible.

This document describes the technical and business aspects of the interface that need to be known by the commercial banks so that they can implement the interface and integrate it within their existing IT environment. Note that this is a service available in NG-RTGS, not a tool offered to the Participants. So each bank needs to develop its own implementation according to the specifications laid in this document. Also, Montran/TCS can offer alternative implementation solutions of the Application Programming Interface (API) interface to those Participants that are interested, but outside the scope of this project.

2. Business Aspects

2.1. Overall Architecture

The most important characteristics of the NG-RTGS Web service interface are as follows:

a) It is an online, real-time interface
b) It conveys payment messages to and from the NG-RTGS in the format supported by the NG-RTGS
c) It is based on a request-response protocol
d) It is based on open standards which can be implemented equally on several different technical platforms (for example Java, C++, C# and so on.)
e) It is secured and the messages are authenticated through digital signatures

The NG-RTGS system can receive payment instructions from the Participants over three main networks, as described in the following diagram:

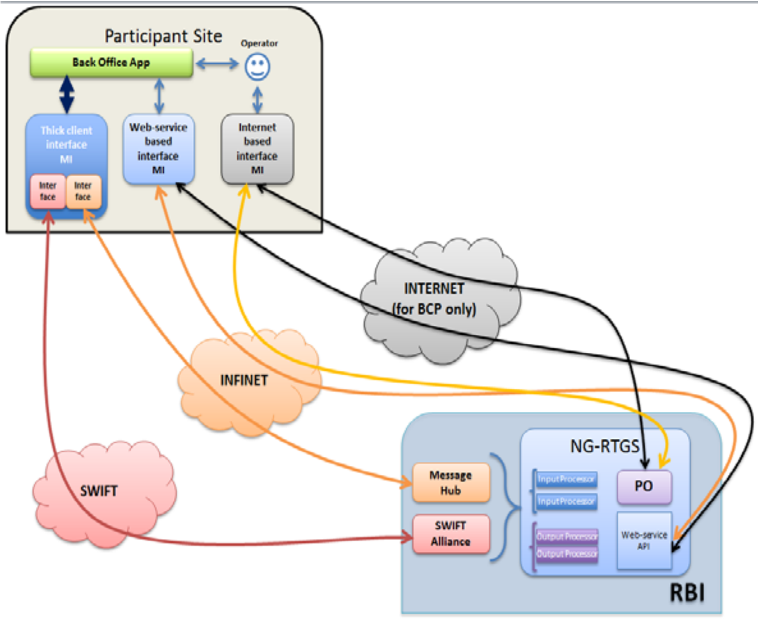

Figure 1: Network Flows

The SWIFT and INFINET networks are considered the main channels of communication with the NG-RTGS for the majority of the commercial banks. Therefore, the Web service interface should be considered as a contingency solution for those moments when the main Straight-Through Processing (STP) interface offered by SWIFT and Institute for Development & Research in Banking Technology (IDRBT) is not available. A typical flow of operations over the Web service interface is as follows:

1. The main STP interface (SWIFT/INFINET) stops working for Participant A.
2. The Participant notifies RBI to update the bank's profile in NG-RTGS to reflect the fact that any further messages for the Participant should not be send anymore to the default STP channel.
3. The Participant activates its application that implements the Web service interface and uses it to connect to the NG-RTGS through this interface. Message exchange is now carried on over the Web service interface.
4. After the main STP connectivity is recovered, the RBI updates the Participant profile and the NG-RTGS resumes the sending of the messages over the main channel.

The architecture of the solution follows the typical client-server model. The client application is always the initiator of the communication, sending a request to the central NG-RTGS. Then the NG-RTGS will respond with the requested data (if available), completing the cycle.

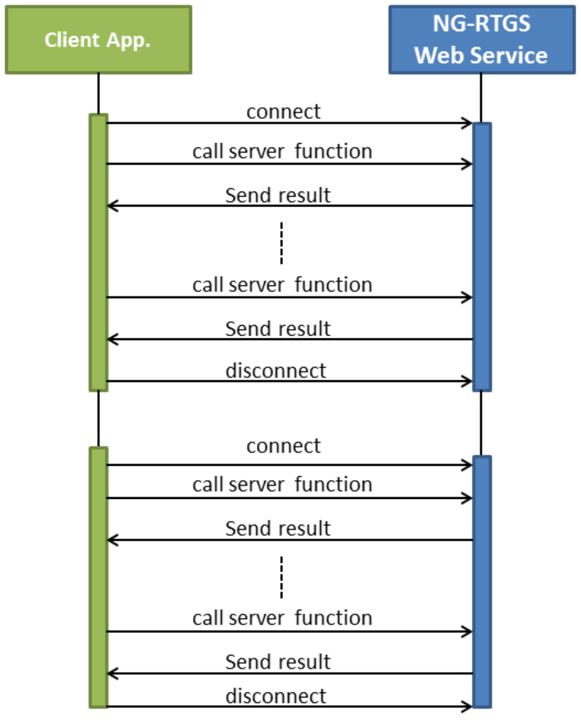

Figure 2: Communication Cycle

The communication between the client application and the central system is secured using the standard Secure Socket Layer (SSL) protocol. The certificate used by the application authenticates the sender as the NG-RTGS will check whether the sending institution mentioned inside the submitted messages is the same with the owner of the certificate.

2.2. Supported Functions

The API exposed by the NG-RTGS system gives the Participants a basic message management which allows them to continue their business with regards to the flow of payment instructions to and from the central settlement system. Below is the list of function calls the Participant's client application can invoke after connecting to the Web service of NG-RTGS.

2.2.1. Function Call: putMessage()

The signature:

```
StringputMessage(Stringmessage);
```

Purpose:

This call allows the sending client application to send a new single payment message to the NG-RTGS. The payment should be in the same format as the messages normally delivered to the INFINET or SWIFT networks. The message should include the business application header and the signature for the business message. For each invocation of this call, the central NG-RTGS system will respond with a result code indicating the result of the upload operation. This should not be confused with the acknowledgment message generated by the NG-RTGS after the processing of the payment instruction submitted over the interface. This is a different message which can be retrieved by the client application using the function call getMessage(). The result code returned by the NG-RTGS merely indicates that the transmitted data was retrieved entirely.

The client application can submit payment for the current business date or for future business dates. If the messages indicate an invalid date (e.g. past value date, past cutoff times etc.) the messages are accepted over this interface but they will fail the business validations of the NG-RTGS. Hence, the NG-RTGS will generate appropriate negative acknowledgement outward messages which can be retrieved by using the getMessage() function call.

The messages submitted to the NG-RTGS must meet the security requirements of the NG-RTGS. For full details please see the security section of this document (3.2).

Arguments:

- message: this represents the message sent to the NG-RTGS application.

Return value:

“0” - for success;
“1” – for invalid data (e.g. null argument)
“2” – internal server error (which should be reported)

2.2.2. Function Call: getMessage()

Purpose:

The client application should call this function in order to retrieve a message from the central NG-RTGS. The caller can either specify the UTR of the last message received from the NG-RTGS or null. If a UTR is provided, the NG-RTGS will respond with the next available outgoing message generated for the respective Participant, if any. If the caller does not specify a starting UTR, the NG-RTGS will respond with the first outward message generated for the current business date. The messages generated by the NG-RTGS will be digitally signed with the certificate of the NG-RTGS. The messages sent by other Participants will include signatures generated by the respective banks.

Signature:

```
String getMessage(Stringutr);
```

Arguments:

- utr: the UTR of the message prior to the message returned by the function call, if any. If null, the system returns the first outgoing message generated for the calling Participant, if any.

Return value:

The actual message in full, including the signature - for success;
“NO_MSG” – for no message found
“2” – internal server error (which should be reported)

2.2.3. Function Call: getBusinessDate()

Purpose:

This function informs the client application about the current business date of the NG-RTGS system. Please note that the NG-RTGS may have a different business date for different currencies. The client application should never send payment messages with a past value date as they will be rejected.

Signature:

```
- String getBusinessDate(String currency);
```

Arguments:

- currency: the ISO code of the currency for which the caller requests the business date. (e.g. INR)

Return value:

- the value of the business date in yyyyMMdd format(e.g. 20130429) - for success;

“1” – for invalid data (e.g. invalid currency)

“2” – internal server error (which should be reported)

2.2.4. Function Call: getCurrentTimetable()

Purpose:

To assist the Participants in sending their payments within the correct timeframe, the NG-RTGS can report the current situation of the system’s timetable, namely the name and status of the time events which controls the submission of various types of payments for settlement. The response will be formatted as an XML document, as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<timetable>

        <system>NGRTGS</system>

        <bdate>20130303</bdate>

        <curr>INR</curr>

        <content>

                <event name="SOD" mode="MANUAL" status="EXECUTED" />

                <event name="OFB" mode="AUTO" status="EXECUTED" startTime="09:00" />
```

```
                        <event name="IC" mode="AUTO" status="EXECUTED" startTime="16:00" />
                        <event name="FC" mode="AUTO" status="EXECUTED" startTime="17:00" />
                        <event name="IDL" mode="AUTO" status="EXECUTED" startTime="17:15"
endTime="18:00" />
                        <event name="EOD" mode="MANUAL" status="PRIOR" startTime="18:00" />
            </content>
</timetable>
```

The equivalent XSD schema is presented below:

```
<xs:schemaelementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="timetable">
<xs:complexType>
<xs:sequence>
<xs:element type="xs:string" name="system"/>
<xs:element type="xs:string" name="bdate"/>
<xs:element type="xs:string" name="curr"/>
<xs:element name="content">
<xs:complexType>
<xs:sequence>
<xs:element name="event">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute type="xs:string" name="name" use="required"/>
            <xs:attribute type="xs:string" name="mode" use="required"/>
            <xs:attribute type="xs:string" name="status" use="required"/>
<xs:attribute type="xs:string" name="startTime" use="optional"/>
<xs:attribute type="xs:string" name="endTime" use="optional"/>
</xs:extension>
</xs:simpleContent>
```

```
</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>
```

The following conventions are implied:

Node	Value	Description
system	NGRTGS	It contains the name of the application (NGRTGS). It's a fix string.
bdate	20130429	It reflects the current business date of the NG-RTGS system. The format of the information is: yyyyMMdd (e.g. 20130303)
curr	INR	It contains the ISO code of the currency this timetable is for. Note that the NG-RTGS application maintains different timetables for different currencies.
name	- SOD - OFB - IC - FC - IDL - EOD	This code represents the name of the time event present in the timetable. The codes stand for: SOD: start of the day OFB: open for business IC: Initial Cut-off FC: Final Cut-off IDL: Final IDL Session EOD: End of Day
mode	- AUTO - MANUAL	It indicates whether the current event is scheduled to be

		performed automatically or an operator is going to manually perform it. For manual events, no execution times will be provided.
status	- PRIOR: the event hasn't been performed yet - EXECUTING: the event is currently executing - EXECUTED: the event has been performed successfully - ACTIVE: it indicates that the current session event is currently undergoing, where the starting time was passed but the ending time hasn't been reached yet. It applies only to those events that have a starting and a ending timing (i.e. a session). - ENDING: used when a session event is currently in the process of closing.(e.g. It reached the ending time)	It indicates the current status of the event
startTime		Indicates the time when the event is scheduled to be performed. The format of the information is: hh:mm. (E.g. 08:05). It's not present for manual events.
endTime		Indicates the time when a session event is supposed to end. The format of the information is: hh:mm

Signature:

```
String getCurrentTimetable(String currency);
```

Arguments:

- currency: the ISO code of the currency for which the caller requests the timetable. (e.g. INR).

Return value:

The respective XML message - for success;
"1" – for invalid argument
"2" – internal server error (which should be reported)

3. Technical Aspects

3.1. The Web Service Parameters

The web-service provided by NG-RTGS is based on the SOAP protocol with HTTPS as transport. The low-level details of the service are provided below.

The WSDL XML definitions:

```
- <?xml version='1.0' encoding='UTF-8'?><definitions xmlns:wsu="http://docs.oasis-
  open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://ws.message.impl.tp.rtgs.montran.com/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://ws.message.impl.tp.rtgs.montran.com/"
  name="WSMessageGatewayImplService">
- <wsp:Policy wsu:Id="WSMessageGatewayImplPortBinding_MTOM_Policy-
  WSMessageGatewayImplPortBinding_MTOM_Policy">
- <ns1:OptimizedMimeSerialization
  xmlns:ns1="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization"
  wsp:Optional="true"/>
- </wsp:Policy>
- <types>
- <xsd:schema>
- <xsd:import namespace="http://ws.message.impl.tp.rtgs.montran.com/"
  schemaLocation="http://localhost:7081/WSMessageGatewayImplService/WSMessageGatew
  ayImpl?xsd=1"/>
- </xsd:schema>
- </types>
- <message name="getMessage">
- <part name="parameters" element="tns:getMessage"/>
- </message>
- <message name="getMessageResponse">
- <part name="parameters" element="tns:getMessageResponse"/>
- </message>
- <message name="getCurrentTimetable">
- <part name="parameters" element="tns:getCurrentTimetable"/>
- </message>
- <message name="getCurrentTimetableResponse">
- <part name="parameters" element="tns:getCurrentTimetableResponse"/>
- </message>
- <message name="putMessage">
- <part name="parameters" element="tns:putMessage"/>
- </message>
- <message name="putMessageResponse">
- <part name="parameters" element="tns:putMessageResponse"/>
- </message>
- <message name="getBusinessDate">
- <part name="parameters" element="tns:getBusinessDate"/>
- </message>
- <message name="getBusinessDateResponse">
-    <part name="parameters" element="tns:getBusinessDateResponse"/>
-    </message>
-    <portType name="WSMessageGatewayImpl">
-    <operation name="getMessage">
-    <input
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getMes
     sageRequest" message="tns:getMessage"/>
-    <output
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getMes
     sageResponse" message="tns:getMessageResponse"/>
-    </operation>
-    <operation name="getCurrentTimetable">
-    <input
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getCurr
     entTimetableRequest" message="tns:getCurrentTimetable"/>
-    <output
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getCurr
     entTimetableResponse" message="tns:getCurrentTimetableResponse"/>
-    </operation>
-    <operation name="putMessage">
-    <input
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/putMes
     sageRequest" message="tns:putMessage"/>
-    <output
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/putMes
     sageResponse" message="tns:putMessageResponse"/>
-    </operation>
-    <operation name="getBusinessDate">
-    <input
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getBusi
     nessDateRequest" message="tns:getBusinessDate"/>
-    <output
     wsam:Action="http://ws.message.impl.tp.rtgs.montran.com/WSMessageGatewayImpl/getBusi
     nessDateResponse" message="tns:getBusinessDateResponse"/>
-    </operation>
-    </portType>
-    <binding name="WSMessageGatewayImplPortBinding"
     type="tns:WSMessageGatewayImpl">
-    <wsp:PolicyReference URI="#WSMessageGatewayImplPortBinding_MTOM_Policy-
     WSMessageGatewayImplPortBinding_MTOM_Policy"/>
-    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
-    <operation name="getMessage">
-    <soap:operationsoapAction=""/>
-    <input>
-    <soap:body use="literal"/>
-    </input>
-    <output>
-    <soap:body use="literal"/>
-    </output>
-    </operation>
-    <operation name="getCurrentTimetable">
-    <soap:operationsoapAction=""/>
-    <input>
-    <soap:body use="literal"/>
```

- </input>
- <output>
- <soap:body use="literal"/>
- </output>
- </operation>
- <operation name="putMessage">
- <soap:operationsoapAction=""/>
- <input>
- <soap:body use="literal"/>
- </input>
- <output>
- <soap:body use="literal"/>
- </output>
- </operation>
- <operation name="getBusinessDate">
- <soap:operationsoapAction=""/>
- <input>
- <soap:body use="literal"/>
- </input>
- <output>
- <soap:body use="literal"/>
- </output>
- </operation>
- </binding>
- <service name="WSMessageGatewayImplService">
- <port name="WSMessageGatewayImplPort" binding="tns:WSMessageGatewayImplPortBinding">
- <soap:address location="http://localhost:7081/WSMessageGatewayImplService/WSMessageGatewayImpl"/>
- </port>
- </service>
- </definitions>

The XSD schema definitions:

- <?xml version="1.0" encoding="UTF-8"?>
- <xs:schemaxmlns:tns="http://ws.message.impl.tp.rtgs.montran.com/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0" targetNamespace="http://ws.message.impl.tp.rtgs.montran.com/">
-
- <xs:element name="getBusinessDate" type="tns:getBusinessDate" />
-
- <xs:element name="getBusinessDateResponse" type="tns:getBusinessDateResponse" />
-
- <xs:element name="getCurrentTimetable" type="tns:getCurrentTimetable" />
-
- <xs:element name="getCurrentTimetableResponse" type="tns:getCurrentTimetableResponse" />
-
- <xs:element name="getMessage" type="tns:getMessage" />
-
- <xs:element name="getMessageResponse" type="tns:getMessageResponse" />
-

```
-    <xs:element name="putMessage" type="tns:putMessage" />
-
-    <xs:element name="putMessageResponse" type="tns:putMessageResponse" />
-
-    <xs:complexType name="getBusinessDate">
-    <xs:sequence>
-    <xs:element name="arg0" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="getBusinessDateResponse">
-    <xs:sequence>
-    <xs:element name="return" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="putMessage">
-    <xs:sequence>
-    <xs:element name="arg0" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="putMessageResponse">
-    <xs:sequence>
-    <xs:element name="return" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="getCurrentTimetable">
-    <xs:sequence>
-    <xs:element name="arg0" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="getCurrentTimetableResponse">
-    <xs:sequence>
-    <xs:element name="return" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="getMessage">
-    <xs:sequence>
-    <xs:element name="arg0" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-
-    <xs:complexType name="getMessageResponse">
-    <xs:sequence>
-    <xs:element name="return" type="xs:string" minOccurs="0" />
-    </xs:sequence>
-    </xs:complexType>
-    </xs:schema>
```

The service will be available over any available network and the communication will be secured by using SSL encryption based on digital certificates issued by RBI to each Participant.

3.2. The Security Requirements

The security requirements at both transport and message levels are provided in the below section.

3.2.1. The Transport Level

The client application must use the HTTPS protocol to connect to the NG-RTGS system. This requires a server level digital certificate that was issued by IDRBT for the respective Participant. The serial number of the certificate must be mapped to respective participant in RTGS system to prevent the submission of messages by other banks.

3.2.2. The Message Level

Every payment instruction message received by NG-RTGS or sent by NG-RTGS must have a digital signature for non-repudiation and authenticity. The signature will be generated using the digital certificate of the system that generated the message.

The format of the signed message must comply with Cryptographic Message Syntax Standard (PKCS#7) with detached signatures. This means the envelope will contain only the certificate and the signature, but not the actual data.

The algorithm to be used is **RSAwithSHA256** with RSA keys of **2048** bit.

The messages sent and received over this interface will be signed but unencrypted as the encryption takes place at the transport level being provided by the HTTPS protocol.