Confidential and Verifiable Email

IncaMail

Enterprise Application Integration Guide

Description for integrating IncaMail functionality into third-party applications

Version: 1.27
Date: April 11, 2024
Author: Post CH Communication Ltd.

Copyright Post CH Communication Ltd.

Table of Contents

Copyright Post CH Communications Ltd

Copyright Post CH Communications Ltd

1. Overview

1.1 Introduction

The confidential and verifiable email service IncaMail runs as a web application, a mobile app or built into standard email clients (Outlook etc.). Additionally, the functionality of IncaMail can be integrated into enterprise applications (e.g. SAP®). .

1.2 Goal and Scope of This Document

This document specifies the four main methods for integrating IncaMail into enterprise applications and provides all necessary information for developers for setup, test and implement an interface to the IncaMail service.

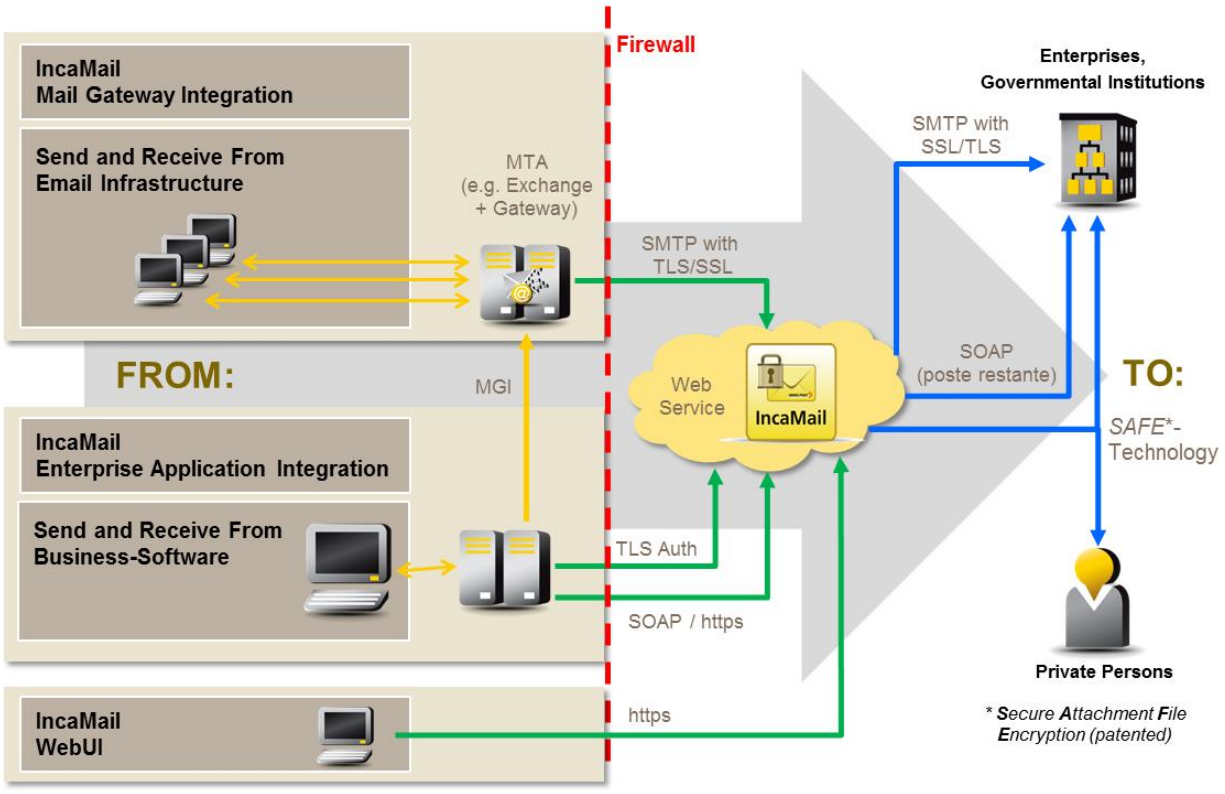
For using the SOAP or REST APIs, developers must be familiar with the principals of SOAP and REST in their own programming language.

1.3 The Four Integration Methods

For integrating IncaMail's functionality into enterprise applications, one of four methods may be choosen. The choice depends on the desired functionality and the flexibility of the third party software.

Method	Description	Functionality
SMTP/TLS Auth	In this simplest form, the IncaMail server is available as a standard SMTP mail server with TLS authentication. It is used for sending emails only.	Send any emails to any recipient using the delivery types „confidential", „personal" or „registered".
Domain Integration (SMTP over TLS/SSL)	Secure connection of a mail domain using a corporate mail server.	Same as TLS Auth, but mail server can store sent messages in mailboxes.
SOAP: Send, Read & Admin	Three powerful SOAP API's for best flexibility and performance. They can also be used to extend other integration methods by trackability etc.	**SOAP Send**: Send any emails to any recipient using the delivery types „confidential", „personal" or „registered". Track the delivery states of every sent message. **SOAP Admin**: Add and modify user accounts to IncaMail **SOAP Read**: Decrypt received encrypted IncaMail messages
REST: Send	Send messages and track their way to the recipient. Use MIME or Large File attachments.	**REST Send:** Send any emails to any recipient using the delivery types "signed-only", „confidential", „personal" or „registered". Track the delivery states of every sent message. Send large files up to 1GB.

Copyright Post CH Communications Ltd



1.4 Requirements

A company using a business software with IncaMail integration has to be a registered IncaMail corporate customer with a valid contract. Additionally, the integration method used by the business software must be activated and configured by the IncaMail support.

1.5 Understanding the patented „SAFE“ and “HYBRID” Formats

The IncaMail service transports encrypted messages by email to *any* email address without the recipient himself having to use PGP or S/MIME encryption technology.

The IncaMail service does this by using one of the following methods:

a) Use full transport-layer encryption (TLS/SSL) to deliver the confidential and/or registered messages
   - directly to registered clients with domain integration or
   - to enterprise applications using SOAP to read cached messages (“poste restante”).
b) Encrypting[1] the complete email message into a message container, the so-called **SAFE attachment**. To decrypt a SAFE attachment, the recipient uses the IncaMail web interface, an IncaMail mobile app or an application with integrated IncaMail functionality.
c) **Hybrid**: Same as b), but adding a “Read” button for faster reception and opening.

The sending business software does not have to be aware how the encryption of the message is done.

By default, IncaMail does never store messages. The full encrypted message is contained in the SAFE attachment instead. It is decrypted or using the web interface or using the SOAP Read interface. However, Hybrid messages leave a double-encrypted copy of the message on the IncaMail server, but the key to decrypt it remains in the Hybrid message itself.

[1] Encryption is done using standard and proven cryptographic algorithms

Copyright Post CH Communications Ltd

2. SMTP/TLS Auth

2.1 Overview

The IncaMail service can behave like an “ordinary” SMTP MTA (message transfer agent or mail transfer agent) and used to send emails using the standard SMTP protocol. Additional tags are included into the email’s subject line to instruct IncaMail about the delivery type to be used.

This way standard functionality for sending emails can be used in any programming language.

2.2 Requirements

The company using a business software with IncaMail integrated using TLS Auth must be registered with IncaMail for a valid corporate customer contract with EAI option for TLS Auth. This must have been configured by the IncaMail support or sales team before the API can be used.

The port being used (587) must not be blocked by the firewall and therefore be registered there.

2.3 Access Configuration

SMTP Production Server	If mx lookup is supported: **`incamail.com`** If mx lookup is not supported: **`gw1.incamail.com`**
SMTP Integration Server (for development and testing)	If mx lookup is supported: **`sta.incamail.com`** If mx lookup is not supported: **`gw1.sta.incamail.com`**
Authentication	Password normal (Basic Authentication)
Port	25 or 587
Username	Registered email address for TLS Auth
Password	Self-defined during registration

NOTE: STARTTLS must be used to switch to TLS encryption prior to sending the password. Without TLS, the communication to IncaMail is not protected!

NOTE: The email address used for authentication must be the same as the sender’s email address in the `FROM:` header line of the message.

NOTE: The message must be formatted in a correct MIME format. If html is used, we recommend to add a text part as well (content-type: multipart/alternative).

Copyright Swiss Post

NOTE: The TLS Auth API requires knowledge about proper MIME message structuring. Using the SOAP API instead can help to avoid this complexity.

2.4 Usage

Before sending an IncaMail message, the following header field has to be modified. It will be used by IncaMail to determine how the message will be delivered:

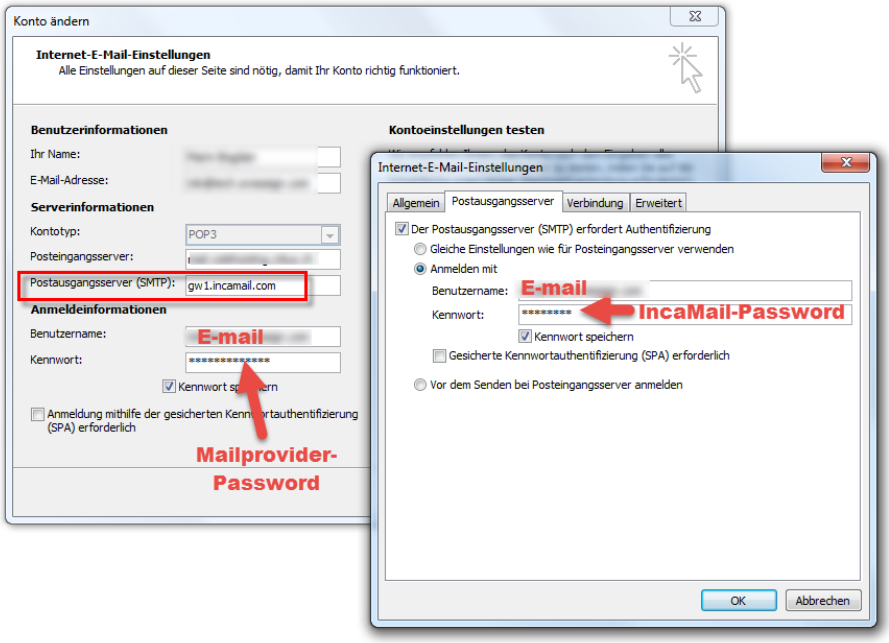
Field	*Description*	*Example*
Subject	Prepend the following tags for IncaMail delivery types: <im> <c> for Confidential emails <im> <p> for Personal emails <im> <r> for Registered emails	`<im> <c>Salary Slip 12/2019`

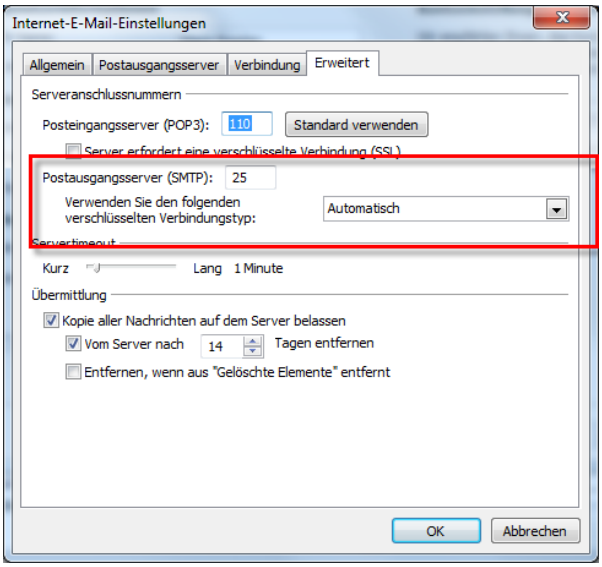
2.5 Example: Use SMTP/TLS Auth in Microsoft® Outlook®

Microsoft® Outlook® can be configured for testing TLS Auth with IncaMail. To do this, create a new email account and change the Outgoing Server options as follows. The Incoming Server settings are used as defined by the email service provider. It is important to have the incoming settings configured correctly for two reasons:

a) Error messages will be sent to the email address and must be read.
b) Microsoft® Outlook® requires to set up both incoming and outgoing servers.

Account Settings:





Of course this configuration could also be used to send IncaMail messages over MAPI, but we recommend to use the SOAP API instead.

3. Domain Integration

3.1 Overview

Domain integration is used in the following cases:

- IncaMail with domain integration is already used in a company's mail infrastructure and additionally, a business software has to use the system.
- Outbound email has to be stored on the company's mail server for archiving purposes.
- The business software uses the company's mail server anyway for sending emails.

Please follow IncaMail Domain Integration documentation for setting up the system.

Please take a look at the video, which explains the concept of mail server integration
→Video

For tracking sent messages, or the IncaMail web interface is used or the SOAP API is called.

3.2 Requirements

Domain integration has been set up for the company by IncaMail support including contract, configuration and testing.

3.3 Usage

Use the mail server's email sending API for sending emails. Before sending an IncaMail message, the following header fields have to be modified. They will be used by special rules on the MTA which make sure the email is sent directly to the IncaMail server over a secured connection:

Field	*Description*	*Example*
Subject	Prepend the following tags for IncaMail delivery types: <im> <c> for Confidential emails <im> <p> for Personal emails <im> <r> for Registered emails	`<im> <c>Salary Slip 12/2019`
To	Append incamail.ch to every email address.	`john.doe@gmail.com.incamail.ch, mary.doe@yahoo.com.incamail.ch`
CC	Append incamail.ch to every email address.	
BCC	Append incamail.ch to every email address.	

The corporate MTA will make sure the message is delivered over the IncaMail service.

Copyright Swiss Post

4. SOAP API

4.1 Goal

The SOAP API (application programming interface) enables the functionality of IncaMail to be integrated in business applications.

This chapter addresses developers who want to add IncaMail functionality to their business applications by using SOAP. They learn the scope of services, how to quickly run and test examples, and how to call IncaMail services from within their applications.

4.2 Scope

4.2.1 One System - Three APIs

The IncaMail SOAP API is divided into three parts:

API	*Description*
SOAP Send	Send any emails to any recipient using the delivery types „confidential“, „personal“ or „registered“. Track the delivery states of every sent message. This API is compatible with its predecessor, SOAP in IncaMail Release 3.
SOAP Read	• decrypt received encrypted IncaMail messages • use a “ping” test to verify SOAP connectivity
SOAP Admin	Modify Corporate Customer Account Data

Every API has its own WSDL definition.

The sender becomes authenticated via user name and password.

4.2.2 SOAP Send: Send and Track Messages

Operation	*Description*
SendMessage	Send an email over the IncaMail service with or without attachments in one of the delivery types Confidential, Personal or Registered. Returns a unique ID for this message.
GetDeliveryStatus	Track the delivery status of a specific message using the ID
GetDeliveryStates	Track the delivery status of a list of messages specified by their IDs

4.2.3 SOAP Read: Decrypt Messages

SOAP Read offers an operation for decoding SAFE attachments stored on a mail server whenever required.

Operation	*Description*
Ping	Test the availability of the IncaMail SOAP service and the correct configuration of the SOAP client
SearchMessages	DEPRECATED
ReadMessage	• Decrypt a message which is available in encrypted form (SAFE)

4.2.4 SOAP Admin: Modify Corporate Customer Account Data

Operation	*Description*
getCustomer	Get the company information currently stored for the SOAP user
mergeCustomer	Modify banner and logo can be used for a consistent CI (Corporate Identity) of the sender and improve the recipient's trust in IncaMail messages.

Copyright Swiss Post

4.3 Authentication

A users of the SOAP interface have to by authorized explicitly to use this service via user name and password using *Preemptive Basic Authentication*.

The connection must be established via a secure https connection. If the credentials are not correct, a http exception is thrown.

Example: Basic Authentication	

JAVA	`final URL url = new URL(endPointUrl);`
	`connection = (HttpURLConnection) url.openConnection();`
	`/**`
	`* HTTP authentication`
	`* String encoded = Base64.encode(username+":"+password);`
	`* connection.setRequestProperty("Authorization", "Basic "+encoded);`
	`*`
	`* Base64.encode(String) would be some method to encode a string in Base 64.`
	`* Yes, that's all you have to do in order to use Basic Auth.`
	`* The code above to set the Request Property should be done immediately`
	`* after opening the connection and before getting the Input or Output streams.`
	`*/`
	`connection.setRequestProperty(HttpHeaders.AUTHORIZATION, "Basic " + basicUtil.encode(username, password));`

4.4 Download IncaMail Server SSL Certificate

Some business applications (e.g. SAP®) require the root SSL certificate of the IncaMail server to be downloaded and registered in the certstore to be able to use the SOAP API.

The actual public root certificate can be downloaded here: https://www.swisssign.com/support/ca-prod.html

> ***NOTE:*** The old root certificate “Server Gold G22” is not used any more after May 29, 2023, and must be replaced by “SwissSign Gold CA – G2” or “Root CA: Gold G2”.

SwissSign Gold CA - G2 / Root CA: Gold G2	
Key-ID	5B:25:7B:96:A4:65:51:7E:B8:39:F3:C0:78:66:5E:E8:3A:E7:F0:EE
Serial	BB:40:1C:43:F5:5E:4F:B0
Fingerprint (SHA2)	62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:95

Copyright Swiss Post

Download Certificate	
PEM	https://swisssign.net/cgi-bin/authority/download/5B257B96A465517EB839F3C078665EE83AE7F0EE.pem
DER	https://swisssign.net/cgi-bin/authority/download/5B257B96A465517EB839F3C078665EE83AE7F0EE

Revocation List	
PEM	https://crl.swisssign.net/5B257B96A465517EB839F3C078665EE83AE7F0EE.pem
DER	https://crl.swisssign.net/5B257B96A465517EB839F3C078665EE83AE7F0EE

4.5 SOAP Throughput Optimization

The throughput (SOAP messages sent per hour) vary depending on several factors:

- IncaMail Server load
- SOAP Client (Enterprise Application) internet connection upload speed and load
- SOAP Client (Enterprise Application) performance and load
- …

The IncaMail service is built to process several SOAP requests at the same time. This can be used to increase the transfer speed by sending them from several threads/processes in parallel.

> ***NOTE:*** We recommend not to use more than four parallel threads for sending SOAP requests to the IncaMail server.

4.6 IncaMail SOAP Service Endpoints, WSDL, XSD

4.6.1 Definitions

SOAP Endpoint	URL, where a SOAP service can be accessed by a client application. IncaMail has one Endpoint per API.
SOAP WSDL	WSDL (Web Services Description Language) files contain an XML-formatted description of a SOAP API, consisting of one or more exposed operations (methods).
SOAP XSD	XML Schema file describing the types used by a SOAP API

4.6.2 Generate Code for Using the IncaMail SOAP API

Most modern programming languages (respectively their development environments) can generate code (classes) for SOAP client functionality directly from a WSDL endpoint interface (see 4.6.3 Access to Production and 4.6.4 Access to Test).

Code generation is different for every programming language and is not covered by this guide. However, the SOAP examples provided can be used for SOAP UI (for a quick start) and can be easily integrated into any generated code.

4.6.3 Access to Production Platform

URL	*Description*
Endpoints: https://ws.incamail.com/3.0/ https://wsread.incamail.com/3.0/IncaMailReadService https://wsmgmt.incamail.com/↘ IncaMailAdminService/IncaMailAdminPort	This URL must be registered as the endpoint in some systems (e.g. SAP®). **IMPORTANT: It is essential not to forget the final "/" in the URL where needed!**
Endpoint interfaces for WSDL: https://ws.incamail.com/3.0/?wsdl https://wsread.incamail.com/3.0/IncaMailReadService/?wsdl https://wsmgmt.incamail.com/↘ IncaMailAdminService/IncaMailAdminPort/?wsdl	This URL enables access to the interface via the HTTPS protocol with password authentication.
Endpoint interfaces for XSD: https://ws.incamail.com/3.0/?xsd=1 https://wsread.incamail.com/3.0/IncaMailReadService/?xsd=1 https://wsmgmt.incamail.com/↘ IncaMailAdminService/IncaMailAdminPort/?xsd=1	This URL contains the type definitions. Use this to find the definition of data types etc. used by the IncaMail SOAP interface.

4.6.4 Access to Test Platform (Integration)

URL	*Description*
Endpoints: https://ws.sta.incamail.com/3.0/ https://wsread.sta.incamail.com/3.0/IncaMailReadService https://wsmgmt.sta.incamail.com/⇘ IncaMailAdminService/IncaMailAdminPort	This URL must be registered as the endpoint in some systems (e.g. SAP®). **IMPORTANT: It is essential not to forget the final "/" in the URL where needed!**
Endpoint interfaces for WSDL: https://ws.sta.incamail.com/3.0/?wsdl https://wsread.sta.incamail.com/3.0/IncaMailReadService/?wsdl https://wsmgmt.sta.incamail.com/⇘ IncaMailAdminService/IncaMailAdminPort/?wsdl	This URL enables access to the interface via the HTTPS protocol with password authentication.
Endpoint interfaces for XSD: https://ws.sta.incamail.com/3.0/?xsd=1 https://wsread.sta.incamail.com/3.0/IncaMailReadService/?xsd=1 https://wsmgmt.sta.incamail.com/⇘ IncaMailAdminService/IncaMailAdminPort/?xsd=1	This URL contains the type definitions. Use this to find the definition of data types etc. used by the IncaMail SOAP interface.

4.7 Getting Started: Explore the API With SoapUI

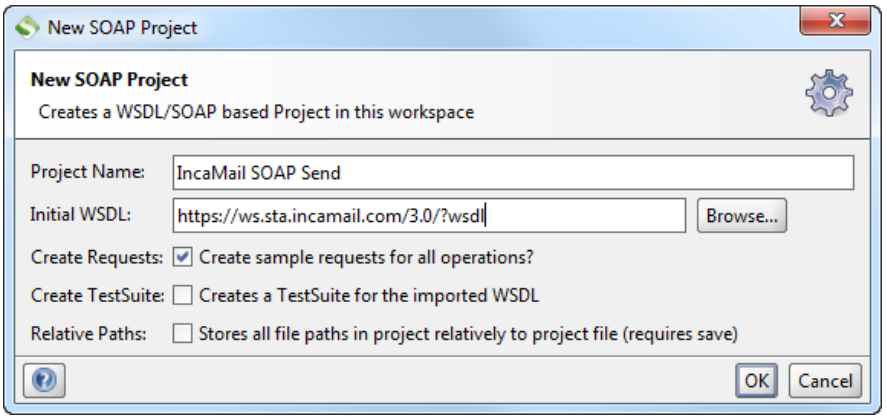
4.7.1 Introduction

The free SOAP tool SoapUI is perfect for doing the first steps with the IncaMail SOAP API. The examples in this guide can directly be used there. SoapUI enables developers to get familiar and test the IncaMail SOAP API without having to write even one line of code.

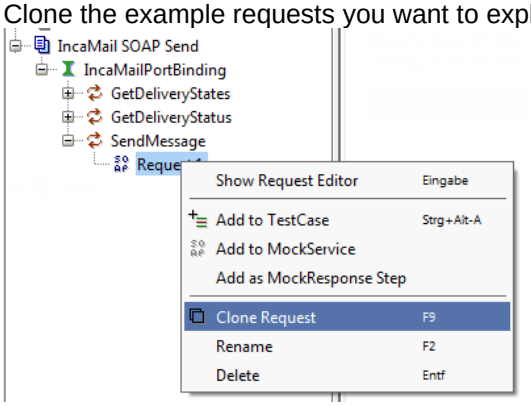
4.7.2 Install and Use

Installation of SoapUI:

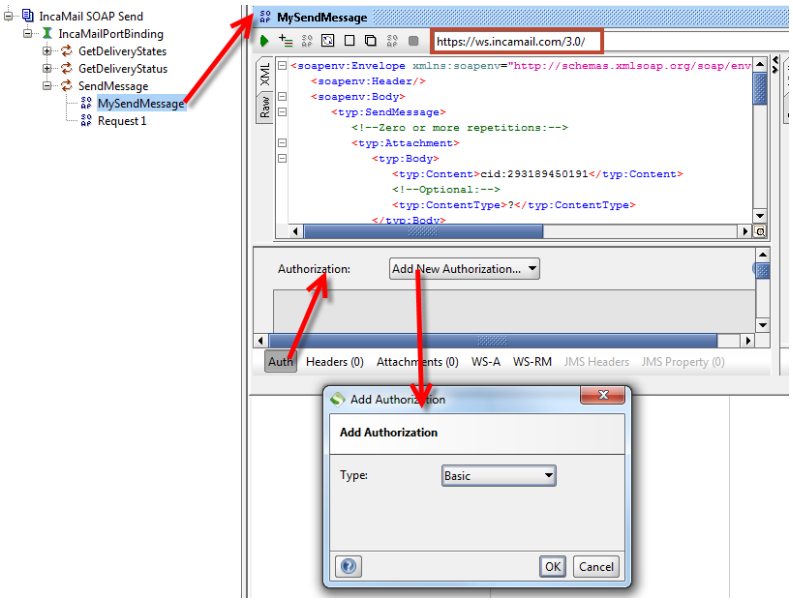
1. Go to http://www.soapui.org/ and install the actual free version of SoapUI
2. Start SoapUI and create a new project using FILE/NEW SOAP PROJECT :



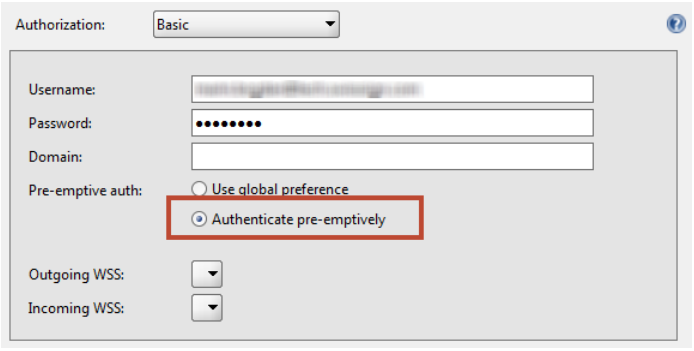
3. Clone the example requests you want to explore and give them a name:



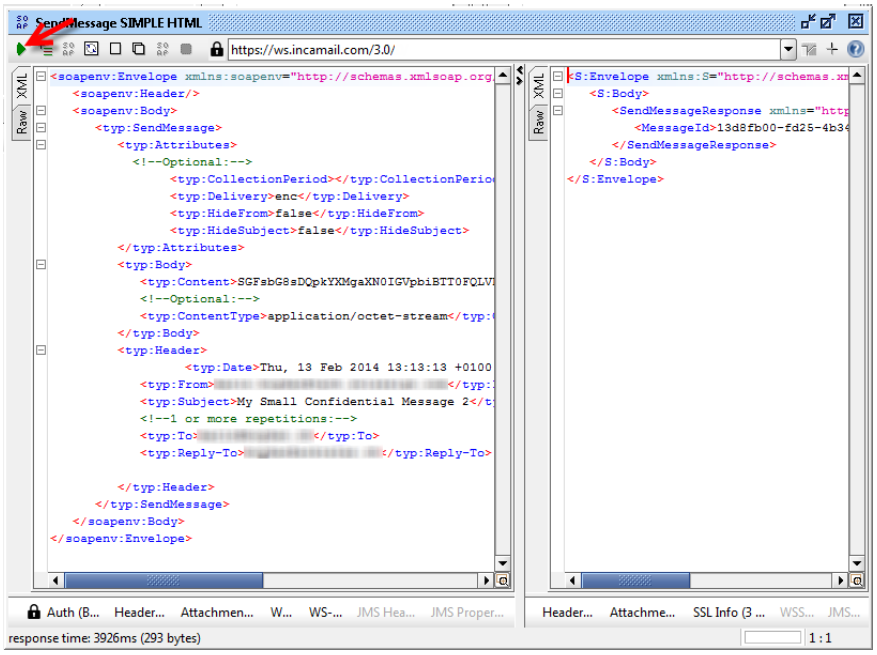
4. Double-clicking the name of the new request will open a window. Enter the correct endpoint for the API you want to test, click on the “Auth” tab in the left lower corner to define the authorization data for this request and select AUTHORIZATION/ADD NEW AUTHORIZATION:



5. Enter you IncaMail account credentials for “Basic Authorization” and “Authenticate pre-emptively”:



6. Modify the SOAP xml request on the left side and press the green arrow to send it:



The result pane on the right side shows the return data from the IncaMail service.

4.8 Operations

4.8.1 SOAP Send API: Send and Track Messages

4.8.1.1 SendMessage

Description

Operation	*Description*
`SendMessage`	Send an email over the IncaMail service with or without attachments in one of the delivery types Confidential, Personal or Registered. Returns a unique ID for this message.

`SendMessage` takes elements of an email like header elements (From, To, Subject…), a message body, attachments and some delivery attributes as parameters, creates standard-conform email messages and sends them to one or more recipients. The IncaMail service is responsible to create, encrypt and sign a valid email message in a format suitable for every recipient and to deliver it. A unique MessageId for this message is returned if the operation was successful. This MessageId can be used to track the message status during the delivery lifecycle using `GetDeliveryStatus` and `GetDeliveryStates`.

NOTE: The SOAP XML requests the UTF8 character set. Avoid using copy & paste texts from text processors using another character set, e.g. Word!

NOTE: The UTF8 character set allows practically any special character without further encoding, e.g. “ü” in the subject line.

However, general XML rules define five special characters which may not be part of the parameters (e.g. “subject”), but must be escaped (replaced by a string):

- ampersand (&) is escaped to &
- double quotes (") are escaped to "
- single quotes (') are escaped to '
- less than (<) is escaped to <
- greater than (>) is escaped to >

Example: The subject Übernachtungen “Weggis” 2017 & 2018 > 2019! should be sent as Übernachtungen "Weggis" 2017 & 2018 > 2019!

NOTE: If you send a HTML body text, use Character Entity References for special characters (e.g. ü for ü): http://dev.w3.org/html5/html-author/charref

> ***NOTE:*** Some parameters must be base64-encoded, e.g. Body/Content. Make sure to use a basic base64-encoding WITHOUT line breaks!

Input Parameters for "SendMessage"

Input Parameter	*Data Type*	*Comment*
Header/Date	Date UTF	Use UTF formatting for the sending datetime, e.g. `Thu, 13 Aug 2019 13:13:13 +0100 (CET)` → Not required! The date is set automatically.
Header/From	String	Sender email address, must be identical to the email used for authorization of the SOAP interface → The API does not allow to send emails in the name of another person for security reasons!
Header/Subject	String	Email Subject → While domain integration and TLS Auth integration need specific tags in the subject, this is not required for SOAP.
Header/To	String	One of several recipient's email addresses, separated by a comma
Header/Cc	String	Optional CC recipient's email addresses, separated by a comma
Header/Bcc	String	Optional BCC recipient's email addresses, separated by a comma
Header/SafeEnvelopeMessage	String	Optional individual text which is shown on the envelope message of an encrypted (SAFE) message. Must be base64-encoded.
Header/Reply-To	String	Email address where replies to the message will be sent to. Can be different from the "From" email address. → This address is only used in the decrypted email, while the Reply-To on the encrypted (SAFE) email always remains the same as the "From" email address.

Input Parameter	*Data Type*	*Comment*
Body/Content	base64 or string	Content of the body as text or html → The body content must be base64-encoded
Body/ContentType	String	MIME Type of Body/Content, e.g. text/html or text/plain

Copyright Swiss Post

Input Parameter	Data Type	Comment
Attachment/Body/ Content	base64	(Binary) content of the attached file
Attachment/Body/ ContentType	String	MIME Type of attachment, e.g. application/pdf, text/html, text/plain etc.
Attachment/Name	String	Name of the attached file including file extension

Input Parameter	Data Type	Comment
Attributes/CollectionPeriod	Number	Depricated
Attributes/Delivery	String	Delivery Type: *Value* — *Description* **enc** — **Confidential** The message is sent encrypted if the recipient does not belong to a TLS domain of IncaMail. **encPersonal** — **Personal** The message is sent encrypted (also if the recipient belongs to a TLS domain of IncaMail). **reco** — **Registered** The message is sent registered. **Receipt (empty)** Receipts do not have a delivery type, while messages always have
Attributes/HideFrom	Boolean	deprecated
Attributes/HideSubject	Boolean	deprecated

Output Parameter	Data Type	Comment
MessageId	String	Unique ID if message has been successfully sent. Can be used to track the delivery state of the message.

The input-message „SendMessage“ contains the elements Attachment, Attributes, Body and Header:

```
<xsd:element name="SendMessage" type="imst:SendMessage"/>
 <xsd:complexType name="SendMessage">
   <xsd:sequence>
     <xsd:element name="Attachment" type="imst:Attachment"
       minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element name="Attributes" type="imst:MessageAttributes"/>
     <xsd:element name="Body" type="imst:MimePart"/>
     <xsd:element name="Header" type="imst:MessageHeader"/>
   </xsd:sequence>
 </xsd:complexType>
 <xsd:element name="SendMessage" type="imst:SendMessage"/>
```

Examples

Example A: Simple Text Message

This is an example of a minimal text message. The text body is sent as plain text of the content type “text/plain”.

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:SendMessage>
         <typ:Attributes>
           <!--Optional:-->
                <typ:CollectionPeriod></typ:CollectionPeriod>
                <typ:Delivery>enc</typ:Delivery>
                <typ:HideFrom>false</typ:HideFrom>
                <typ:HideSubject>false</typ:HideSubject>
         </typ:Attributes>
         <typ:Body>
            <typ:Content>QmVzdCByZWdhcmRzLCANCkluY2FNYWls</typ:Content>
            <!--Optional:-->
            <typ:ContentType>text/plain</typ:ContentType>
         </typ:Body>
         <typ:Header>
                  <typ:Date>Thu, 13 Aug 2019 13:13:13 +0100 (CET)</typ:Date>
            <typ:From>ben.muster@abc.com</typ:From>
            <typ:Subject>Greetings From Switzerland</typ:Subject>
            <!--1 or more repetitions:-->
            <typ:To>Heidi.muster@abc.com</typ:To>
         </typ:Header>
      </typ:SendMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <SendMessageResponse xmlns="http://soap.incamail.ch/3.0/types">
         <MessageId>cc7422b3-509e-4104-b5a4-
0bf431231e8b@im4.signdemo.com</MessageId>
      </SendMessageResponse>
   </S:Body>
</S:Envelope>
```

RAW Request

```
POST https://im4.signdemo.com/3.0/ HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: ""
Authorization: Basic bWFyaW4...bTpNQnRlY2hzd2lzcw==
Content-Length: 1090
Host: incamail.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
```

Example B: Simple HTML Message With Reply-To

This is an example of a simple html message. The text body is sent as an html text (shortened in this example!) of the content type "text/html".

***NOTE**: The html body must be encoded using base64-encoding.*

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:SendMessage>
         <typ:Attributes>
           <!--Optional:-->
                <typ:CollectionPeriod></typ:CollectionPeriod>
                <typ:Delivery>enc</typ:Delivery>
                <typ:HideFrom>false</typ:HideFrom>
                <typ:HideSubject>false</typ:HideSubject>
         </typ:Attributes>
         <typ:Body>
            <typ:Content>SGFsbG8sDQpkYXMgaX...ICBVbHJpY2g=</typ:Content>
            <!--Optional:-->
            <typ:ContentType>text/html</typ:ContentType>
         </typ:Body>
            <typ:Content>DQYqc3...5AB=</typ:Content>
            <!--Optional:-->
            <typ:ContentType>text/plain</typ:ContentType>
         </typ:Body>
         <typ:Header>
            <typ:Date>Thu, 13 Aug 2019 13:13:13 +0100 (CET)</typ:Date>
            <typ:From>ben.muster@abc.com</typ:From>
            <typ:Subject>Greetings From Switzerland</typ:Subject>
            <!--1 or more repetitions:-->
            <typ:To>Heidi.muster@abc.com</typ:To>
            <typ:Reply-To>bmuster@abc.com</typ:Reply-To>

         </typ:Header>
      </typ:SendMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <SendMessageResponse xmlns="http://soap.incamail.ch/3.0/types">
         <MessageId>cc7422b3-509e-5232-b5a4-
0bf431231e8b@im4.signdemo.com</MessageId>
      </SendMessageResponse>
   </S:Body>
</S:Envelope>
```

NOTE: To test the SendMessage request using SoapUI, base64-encode your html or your attachments using one of the online encoder tools, e.g. https://www.base64encode.org/

NOTE: If you send a HTML body text, use Character Entity References for special characters (e.g. ü for ü): http://dev.w3.org/html5/html-author/charref or make sure to use the UTF-8 character set before the base64-encoding.

Copyright Swiss Post

NOTE: Messages with a HTML body should generally be sent with an additional text body for mail previews and for mail clients without HTML support. This additional text body is not automatically generated by IncaMail.

NOTE: The header part of html messages is ignored when being displayed in the IncaMail WebUI (“sanitarisation”).

Example C: HTML Message With Attachments

This is an example of an html message with two pdf attachments.

***NOTE**: The Content of every attachment must be encoded using base64-encoding.*

***NOTE**: Use the content type "application/pdf" and not "application/octet-stream" for pdf files to avoid attachment display problems on some Android devices.*

NOTE: The parameter "Attachment" can be omitted if no attachments are sent.

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:SendMessage>

         <typ:Attachment>
            <typ:Body>
               <typ:Content>JVBERi0xLjMKJcTl8uXrp/...9GCg==</typ:Content>
               <!--Optional:-->
               <typ:ContentType>application/pdf</typ:ContentType>
            </typ:Body>
            <!--Optional:-->
            <typ:Name>SalesResults2018.pdf</typ:Name>
         </typ:Attachment>
         <typ:Attachment>
            <typ:Body>
               <typ:Content>v31ZY1Dccl/0XB/...RU9GCg==</typ:Content>
               <!--Optional:-->
               <typ:ContentType>application/pdf</typ:ContentType>
            </typ:Body>
            <!--Optional:-->
            <typ:Name>SalesResults2017.pdf</typ:Name>
         </typ:Attachment>

         <typ:Attributes>
           <!--Optional:-->
                <typ:CollectionPeriod></typ:CollectionPeriod>
                <typ:Delivery>enc</typ:Delivery>
                <typ:HideFrom>false</typ:HideFrom>
                <typ:HideSubject>false</typ:HideSubject>
         </typ:Attributes>
         <typ:Body>
            <typ:Content>SGFsbG8sDQpkYXMgaX...ICBVbHJpY2g=</typ:Content>
            <!--Optional:-->
            <typ:ContentType>text/html</typ:ContentType>
         </typ:Body>
         <typ:Header>
            <typ:Date>Thu, 13 Aug 2019 13:13:13 +0100 (CET)</typ:Date>
            <typ:From>ben.muster@abc.com</typ:From>
            <typ:Subject>Sales Statistics 2017 &amp;amp; 2018</typ:Subject>
            <!--1 or more repetitions:-->
            <typ:To>Heidi.muster@abc.com</typ:To>
         </typ:Header>
      </typ:SendMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <SendMessageResponse xmlns="http://soap.incamail.ch/3.0/types">
         <MessageId>cc7422b3-509e-4122-b5a4-
0bf431231e8b@im4.signdemo.com</MessageId>
      </SendMessageResponse>
   </S:Body>
</S:Envelope>
```

4.8.1.2 GetDeliveryStatus

Description

Operation	*Description*
GetDeliveryStatus	Track the delivery status of a specific message using the ID

The operation GetDeliveryStatussends back the delivery states for each recipient of a message and for every delivery channel (e.g. “Email” and “Threema message”). The delivery channel is not visible.

Input Parameter for GetDeliveryStatus

Input Parameter	*Data Type*	*Comment*
`MessageId`	String	ID of the message to be tracked, obtained by SendMessage

Output Parameter

Output Parameter	*Data Type*	*Comment*
`Status/Recipient`	String	Email address of recipient
`Status/State`	String	Current delivery status of the message: NOTE: If a message was sent to several recipients, this parameter is issued several times accordingly.
`Status/StateUpdated`	Datetime	Timestamp when the current status has been reached.

State Values for **all** messages:

State	*Logbook Entry in Web Interface*	*Comment*
`new`	Eingetroffen auf IncaMail	Message arrived on IncaMail to be processed
`new`	Eingetroffen auf IncaMail	For registered messages only: this state remains even after the message has been successfully delivered to the recipient’s mail system
`enqueued`	In Warteschlange	Message in waiting queue ready for delivery
`delivered`	Zugestellt	Message has been successfully delivered to the recipient’s mail system (only Confidential and Personal)
`deferred`	Verzögert	Temporary SMTP Error (400): Delivery deferred
`bounced`	Nicht zustellbar	Message can definitely not be delivered (bounce)
`deleted`	Gelöscht (Security)	Message has been deleted due to security problems (virus found)
`displayed`	Geöffnet	The message has been opened (SAFE format).

Copyright Swiss Post

State Values for **registered** messages only:

State	*Logbook Entry in Web Interface*	*Comment*
transferred	Eingetroffen auf Zielplattform	Registered: Message delivered to another delivery platform (eGov)
accepted	Angenommen	Registered: The recipient has accepted the message
auto-accepted	Automatisch angenommen	Registered: The message has automatically been accepted by the recipient's mail system
rejected	Annahme verweigert	Registered: The recipient has refused to accept the message
expired	Verfallen	Registered: Deadline for acceptance passed. Message has not been opened and cannot be opened any more
signature_error	Fehlerhaft (Signatur)	Registered: Invalid message signature
error	Fehlerhaft	Registered: Error when processing a message coming from another delivery platform (eGov)
processed	Verarbeitet	Registered: The message has been processed by IncaMail and is waiting for the recipient

Example

Example: Track the delivery status of a sent message

Use the MessageID which was returned from SendMessage to find the status of a specific message.

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:GetDeliveryStatus>
         <typ:MessageId>2dbfc687-4a88-47cb-bf50-
edb7fe152607@im4.signdemo.com</typ:MessageId>
      </typ:GetDeliveryStatus>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <GetDeliveryStatusResponse xmlns="http://soap.incamail.ch/3.0/types">
         <Status>
            <Recipient>ben.muster@abc.com</Recipient>
            <State>delivered</State>
            <StateUpdated>2019-07-08 21:55:38.781</StateUpdated>
         </Status>
      </GetDeliveryStatusResponse>
   </S:Body>
</S:Envelope>
```

Copyright Swiss Post

4.8.1.3 GetDeliveryStates

Operation	*Description*
GetDeliveryStates	Track the delivery status of list of messages specified by their IDs

The operation GetDeliveryStatessends back the delivery states for a list of sent messages encapsulated in the Element `DeliveryStatusForSentMail` for each sent mail. It works exactly as GetDeliveryStatus, but with a list of MessageIDs as parameters and not a single ID. See documentation for GetDeliveryStatus.

Example

Example: Track the delivery status of a three messages

Use the MessageID which was returned from SendMessage to find the status of a specific message.

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:GetDeliveryStates>
         <!--1 or more repetitions:-->
         <typ:MessageId>c37e2ef7-786f-4be1-9b48@im4.signdemo.com</typ:MessageId>
         <typ:MessageId>4852df44-e29b-463f-9a32@im4.signdemo.com</typ:MessageId>
         <typ:MessageId>476658d9-eaa2-442f-acbe@im4.signdemo.com</typ:MessageId>
      </typ:GetDeliveryStates>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <GetDeliveryStatesResponse xmlns="http://soap.incamail.ch/3.0/types">
         <DeliveryStatusForSentMail>
            <MessageId>c37e2ef7-786f-4be1-9b48@im4.signdemo.com</MessageId>
            <Status>
               <Recipient> ben.muster@abc.com </Recipient>
               <State>delivered</State>
               <StateUpdated>2019-07-09 20:44:18.491</StateUpdated>
            </Status>
         </DeliveryStatusForSentMail>
         <DeliveryStatusForSentMail>
            <MessageId>4852df44-e29b-463f-9a32@im4.signdemo.com</MessageId>
            <Status>
               <Recipient> ben.muster@abc.com </Recipient>
               <State>delivered</State>
               <StateUpdated>2019-07-09 02:02:06.664</StateUpdated>
            </Status>
         </DeliveryStatusForSentMail>
         <DeliveryStatusForSentMail>
            <MessageId>476658d9-eaa2-442f-acbe@im4.signdemo.com</MessageId>
            <Status>
               <Recipient> heidi.muster@abc.com </Recipient>
               <State>delivered</State>
               <StateUpdated>2019-07-06 12:02:25.832</StateUpdated>
            </Status>
         </DeliveryStatusForSentMail>
      </GetDeliveryStatesResponse>
   </S:Body>
</S:Envelope>
```

4.8.2 SOAP Read API: Decrypt Messages

Functionality of the SOAP Read API:

1. Use a Ping operation to verify a connection
2. Use ReadMessage to decrypt an encrypted IncaMail message

> ***NOTE:*** *The former service "Poste Restante" is no longer supported and is not usable with SOAP Read.*

4.8.2.1 Ping

Operation	*Description*
Ping	Test the availability of the IncaMail SOAP service and the correct configuration of the SOAP client

Input Parameter

No input parameter required, only basic authentication.

Output Parameter

Output Parameter	*Data Type*	*Comment*
SubmittedUserName	String	Email address of sender account
HostAddress	String	IncaMail Host IP
HostName	String	IncaMail Host Name

Example

Example: Verify the connection to IncaMail	

Request:	`<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://soap.incamail.ch/3.0/types">`
	`  <soapenv:Header/>`
	`  <soapenv:Body>`
	`  <typ:PingRequest/>`
	`  </soapenv:Body>`
	`</soapenv:Envelope>`

Result:	`<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">`
	`  <S:Body>`
	`  <PingResponse xmlns="http://soap.incamail.ch/3.0/types">`
	`  <SubmittedUserName>ben.muster@abc.com</SubmittedUserName>`
	`  <HostAddress>172.130.20.16</HostAddress>`
	`  <HostName>im43t.test.swisssign.net</HostName>`
	`  </PingResponse>`
	`  </S:Body>`
	`</S:Envelope>`

4.8.2.2 SearchMessages

Operation	*Description*
SearchMessages	DEPRECATED

> ***NOTE:*** *Every IncaMail Message has a unique MessageID and a SafeID. The SafeID is shared among the main message and all receipt messages for this message.*

4.8.2.3 ReadMessage

Operation	*Description*
ReadMessage	• Decrypt a message which is available in encrypted form (SAFE)

Message Decryption

An encrypted IncaMail message is a standard, MIME-compliant email message containing an attachment IncaMail.html. It is used as an envelope to transport an encrypted, original message with optional attachments). When such a message has been received in a mail system mailbox or in a business application, the SOAP API operation ReadMessage can be used to decrypt it.

The attachment IncaMail.html (called “SAFE”) contains html code. Some fields within this code contain values with the complete, encrypted original message and some additional meta data. These values must be extracted to be used as parameters for ReadMessage. The operation will decrypt the message and its attachments and return them. Optionally, the body of the message is rendered as PDF.

Handling of Registered Messages

If the message is a registered message and has not been read before, a receipt is sent to the sender automatically and the message is accepted. This can be avoided by setting RefuseMessage to true. This produces a “rejected” receipt being sent to the sender (and the recipient). In this case, only the message header will be returned without body and attachments. However, timed-out registered messages cannot be read any more.

Input Parameters for ReadMessage

Input Parameter	*Data Type*	*Comment*
SafeId	String	Mandatory: SafeId of the message Can be taken directly from the SAFE (IncaMail.html): The value of the hidden input field “op” contains the SafeId
Recipient	String	Mandatory: Recipient’s email address Can be taken directly from the SAFE (IncaMail.html): The value of the hidden input field “email” contains the Recipient
EncryptedMessage	Base64	Message to be decrypted in Base64-format. The value of the hidden input field “secmail” in the SAFE (IncaMail.html) contains the encrypted message which must be base64-encoded before being used in ReadMessage.
RefuseMessage	Boolean	If this is not set the message is accepted. If it is true, the message is not decrypted, but rejected and cannot read anymore.
IsPostRemaining	Boolean	Always empty or “false” for decrypting messages.
MessageId	String	Always empty
BodyAsPdf	Boolean	Optional: If true, the body part of the message will be returned in a base64 encoded PDF format (instead of a MIME message)

Output Parameters

Output Parameter	*Data Type*	*Comment*
Header/Date	String	Date
Header/From	String	Sender Email
Header/Subject	String	Subject
Header/To	String	Recipient
Body/Content	String	Base64-encoded body of the message (text, html or pdf)
Body/ContentType	String	ContentType of the body
Attachment/Body/Content	String	For each attachment: Base64-encoded content
Attachment/Body/ ContentType	String	For each attachment: Content MIME Type
Attachment/Body/Name	String	For each attachment: Name

Copyright Swiss Post

Examples

Decrypt a SAFE message

The parameters used for this request must be obtained from the attachment IncaMail.html of an encrypted IncaMail message.

Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:typ="http://soap.incamail.ch/3.0/types">
   <soapenv:Header/>
   <soapenv:Body>
      <typ:ReadMessage>
         <!--Optional:-->
<typ:SafeId>525c8b99-a076-4488-8b2a-f19ea0256d7b</typ:SafeId>
<typ:Recipient>heidi.muster@abc.com</typ:Recipient>
         <!--Optional:-->
         <typ:EncryptedMessage>Q29udGVudC1U...BQT09</typ:EncryptedMessage>
         <!--Optional:-->
         <typ:RefuseMessage></typ:RefuseMessage>
         <!--Optional:-->
         <typ:IsPostRemaining></typ:IsPostRemaining>
         <!--Optional:-->
         <typ:BodyAsPdf>false</typ:BodyAsPdf>
      </typ:ReadMessage>
   </soapenv:Body>
</soapenv:Envelope>
```

Result:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ReadMessageResponse xmlns="http://soap.incamail.ch/3.0/types">
         <IncaMessage>
            <SafeId>525c8b99-a076-4488-8b2a-f19ea0256d7b</SafeId>
            <Delivery>enc</Delivery>
            <Header>
               <Date>Fri Jun 3 06:18:11 UTC 2019</Date>
               <From>ben.muster@abc.com</From>
               <Subject>Vertrauliche Verkaufsstatistiken</Subject>
               <To>heidi.muster@abc.com</To>
            </Header>
            <Body>
<Content>SGFsbG8gSGVpZGkNCg0KSWNoIGJpbiBzY2hvbiBmZXJ0aWcgbWl0IGRlbiBTdGF0aXN0aWtlbi
4gQmVhY2h0ZSBkZW4gZXJmcmV1bGljaGVuIFRyZW5kIQ0KDQpHcnVzcw0KQmVu</Content>
               <ContentType>text/plain</ContentType>
            </Body>
            <Attachment>
               <Body>
                  <Content>JVBERi00...KJSVFT0Y=</Content>
                  <ContentType>application/pdf</ContentType>
               </Body>
               <Name>IncaMail_Sales.pdf</Name>
            </Attachment>
         </IncaMessage>
      </ReadMessageResponse>
   </S:Body>
</S:Envelope>
```

4.8.3 SOAP Admin API: Modify Corporate Customer Account Data

4.8.3.1 getCustomer

Operation	*Description*
getCustomer	Read company data of user using SOAP

Input Parameters for getCustomer

There are no input parameters required.

Output Parameters

Output Parameter	*Data Type*	*Comment*
CustomerId	String	Company ID
CompanyName	String	Company Name
Banner/BannerData	Base64	Image
Banner/MimeType	String	Image MIME Type
Banner/BannerFilename	String	Filename of image
Banner/BannerURL	String	Optional: URL of image
CustomerLogo/LogoData	Base64	Image
CustomerLogo /MimeType	String	Image MIME Type
CustomerLogo /BannerFilename	String	Filename of image

Copyright Swiss Post

Example

Example: Read the Current Customer This operation has no parameters. The basic authentication identifies the user and returns data for his company.	
Request:	<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:def="http://soap.incamail.ch/3.0/definition"> <soapenv:Header/> <soapenv:Body> <def:GetCustomer/> </soapenv:Body> </soapenv:Envelope>
Result:	<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:GetCustomerResponse xmlns="http://soap.incamail.ch/3.0/types" xmlns:ns2="http://soap.incamail.ch/3.0/definition"> <GetCustomerResponse> <CustomerData> <CustomerId>c87ff204-e063-4e1b-a2d6-ca032454bd39</CustomerId> <CompanyName>Burlich & Starkimarm Ltd.</CompanyName> </CustomerData> </GetCustomerResponse> </ns2:GetCustomerResponse> </S:Body> </S:Envelope>

4.8.3.2 mergeCustomer

Operation	Description
mergeCustomer	Modify banner and company name used for CI (corporate identity) on IncaMail messages

> ***NOTE:*** *getCustomer has to be called first for obtaining the CompanyId and all other values which in turn are used for mergeCustomer.*

Input Parameters

> ***NOTE:*** *All input values will be changed (if provided) except CustomerId.*

Input Parameter	Data Type	Comment
CustomerId	String	Company ID
CompanyName	String	Company Name
Banner/BannerData	Base64	Image
Banner/MimeType	String	Image MIME Type
Banner/BannerFilename	String	Filename of image
Banner/BannerURL	String	Optional: URL of image
CustomerLogo/LogoData	Base64	Image
CustomerLogo /MimeType	String	Image MIME Type
CustomerLogo /BannerFilename	String	Filename of image

Output Parameters

Output Parameter	Data Type	Comment
CustomerId	String	Date
CompanyName	String	Sender Email

Example

Example: Replace Banner and Company Name	
Request:	<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:def="http://soap.incamail.ch/3.0/definition" xmlns:typ="http://soap.incamail.ch/3.0/types"> <soapenv:Header/> <soapenv:Body> <def:MergeCustomer> <!--Optional:--> <typ:MergeCustomer> <typ:CustomerId>c87ff204-e063-4e1b-a2d6-ca032454bd39</typ:CustomerId> <typ:CompanyName>Burlich & Stronginarm Ltd.</typ:CompanyName> <typ:Banner> <typ:BannerData>iVBORw0...ERW</typ:BannerData> <typ:MimeType>image/png</typ:MimeType> <typ:BannerFilename>MyBanner.png</typ:BannerFilename> <typ:BannerURL></typ:BannerURL> </typ:Banner> <typ:CustomerLogo> <typ:LogoData></typ:LogoData> <typ:MimeType></typ:MimeType> <typ:LogoFilename></typ:LogoFilename> </typ:CustomerLogo> </typ:MergeCustomer> </def:MergeCustomer> </soapenv:Body> </soapenv:Envelope>
Result:	<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"> <S:Body> <ns2:MergeCustomerResponse xmlns="http://soap.incamail.ch/3.0/types" xmlns:ns2="http://soap.incamail.ch/3.0/definition"> <MergeCustomerResponse> <CustomerData> <CustomerId>c87ff204-e063-4e1b-a2d6-ca032454bd39</CustomerId> <CompanyName>Burlich & StronginarmLtd.</CompanyName> <Banner> <BannerData>iVBORw0KGg... TkSuQmCC</BannerData> <MimeType>image/png</MimeType> <BannerFilename>MyBanner.png</BannerFilename> <BannerURL/> </Banner> </CustomerData> </MergeCustomerResponse> </ns2:MergeCustomerResponse> </S:Body> </S:Envelope>

4.9 Fault types

4.9.1 PermissionFault

If a user does not have a subscription or if user and/or password used for authorization are invalid.

```
<xsd:complexType name="PermissionFault">
        <xsd:sequence>
            <xsd:element name="RelatedErrors" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
            <xsd:element name="RelatedWarnings" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
            <xsd:element name="SupportId" type="xsd:string"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:element name="PermissionFault" type="imst:PermissionFault"/>
```

4.9.2 ParameterFault

If an operation fails because an invalid parameter was sent during the request, the interface sends back a fault of the type ParameterFault.

```
<xsd:complexType name="ParameterFault">
<xsd:sequence>
<xsd:element name="RelatedErrors" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
<xsd:element name="RelatedWarnings" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
<xsd:element name="SupportId" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ParameterFault" type="imst:ParameterFault"/>
```

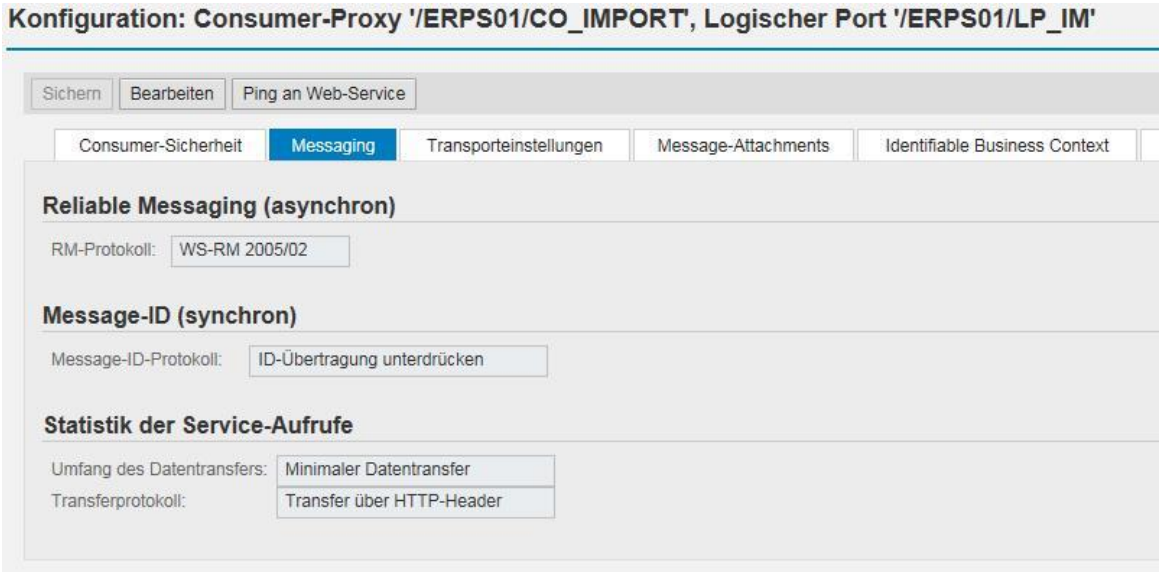
4.9.3 ServiceFault

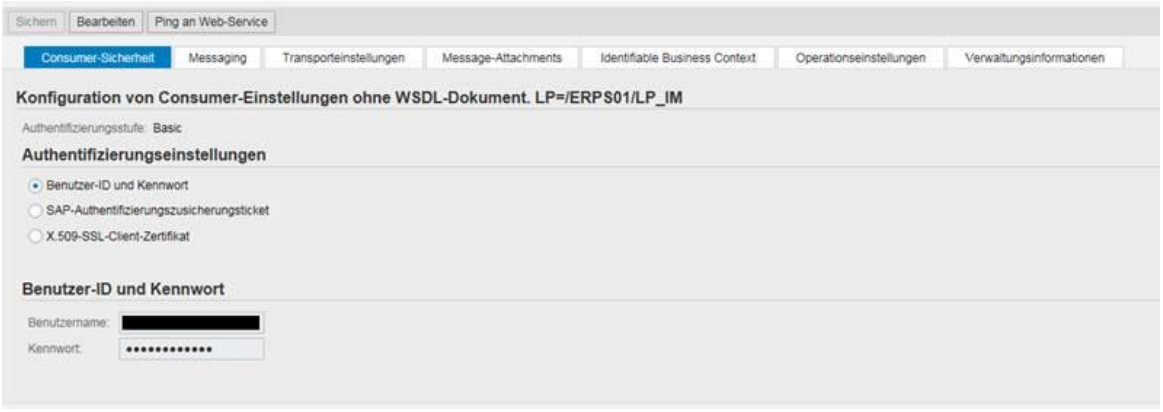
If an internal fault occurs during an operation, the interface sends back a fault of the type ServiceFault.

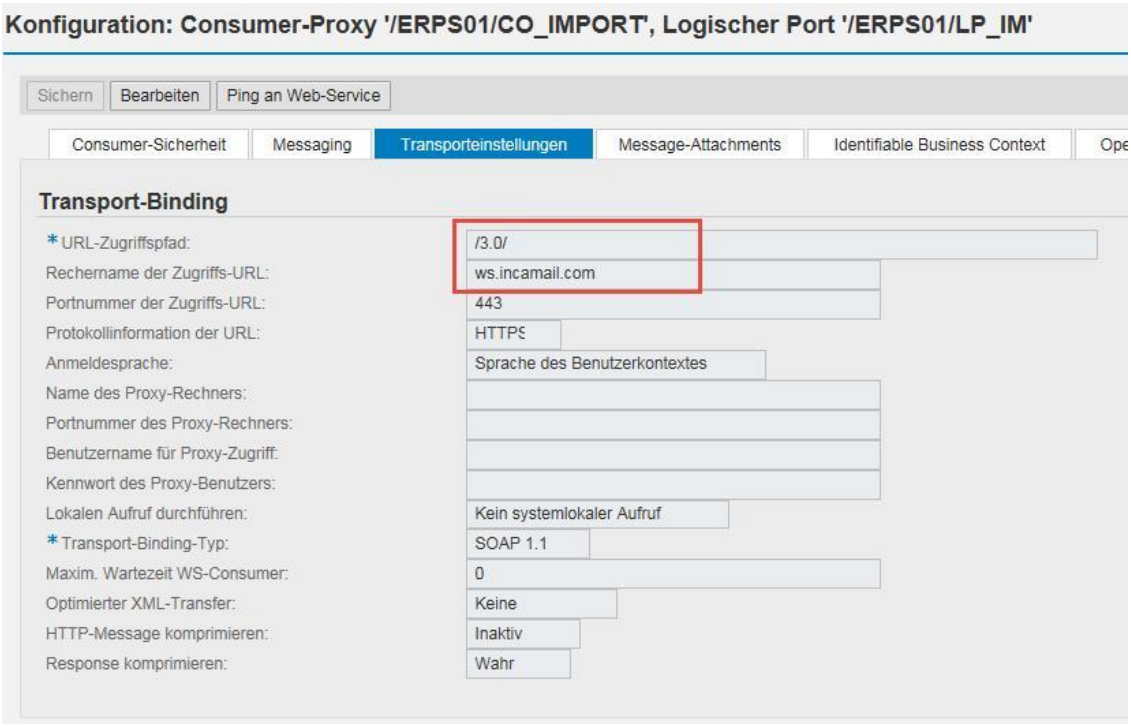
```
<xsd:complexType name="ServiceFault">
<xsd:sequence>
<xsd:element name="RelatedErrors" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
<xsd:element name="RelatedWarnings" type="xsd:string" minOccurs="0"
                                                maxOccurs="unbounded"/>
<xsd:element name="SupportId" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ServiceFault" type="imst:ServiceFault"/>
```

4.10 Setting up SOAP in SAP®

The following settings in SOAPMANAGER have to be set to successfully use SOAP with IncaMail:





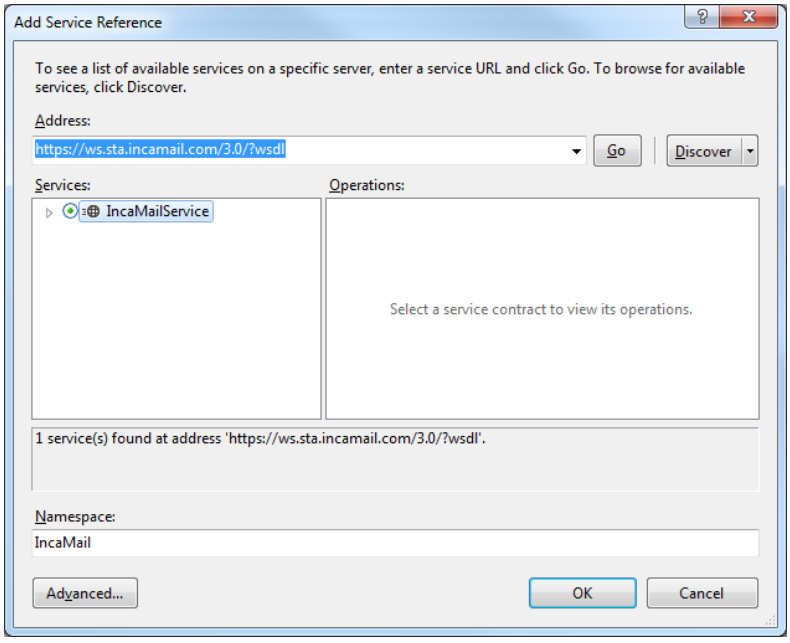


Register the IncaMail server certificate and restart ICM.

4.11 C# Sample Integration

The following is a sample integration of the IncaMail Soap interface in C#.

1.) Add a new Service Reference to your project



This will create all classes in your project to access the interface in your code.

2.) With the creation of the Service Reference, Visual Studio also creates the bindings in either app.config or web.config. Please adapt this binding information:

Binding Information	
app.config or web.config	<system.serviceModel> <bindings> <basicHttpBinding> <binding name="IncaMailPortBinding"> <security mode="Transport" /> </binding> </basicHttpBinding> </bindings> </system.serviceModel>
app.config or web.config *(with proxy)*	<system.serviceModel> <bindings> <basicHttpBinding> <binding name="IncaMailPortBinding" proxyAddress="example.proxy" useDefaultWebProxy="false"> <security mode="Transport" /> </binding> </basicHttpBinding> </bindings> </system.serviceModel>

3.) Create your IncaMail Message, in this example we will create a SendMessage

Create SendMessage object	

C#	var msg = new IncaMail.SendMessage();
	
	// message attributes setup
	msg.Attributes = new MessageAttributes
	{
	Delivery = DeliveryType.enc,
	HideFrom = false,
	HideSubject = false
	};
	
	// message header setup
	msg.Header = new MessageHeader();
	var to = new List<string>();
	var cc = new List<string>();
	var bcc = new List<string>();
	var replyTo = new List<string>();
	
	// add your recipients
	to.Add(„myEmailAddress@test.com“);
	…
	
	// set header
	msg.Header.To = to.ToArray();
	msg.Header.Cc = cc.ToArray();
	msg.Header.Bcc = bcc.ToArray();
	msg.Header.ReplyTo = replyTo.ToArray();
	
	// Setup subject
	msg.Header.Subject = „This is my subject“;

Create SendMessage object

```
// Setup bodies
foreach (var content in ContentList)
{
    string mediaType;
     switch (content.Type)
     {
        case "html":
             mediaType = MediaTypeNames.Text.Html;
             break;
        case "xml":
             mediaType = MediaTypeNames.Text.Xml;
             break;
        case "richtext":
             mediaType = MediaTypeNames.Text.RichText;
             break;
        default:
             mediaType = MediaTypeNames.Text.Plain;
             break;
     }

     var encoding = content.Encoding != null ?
content.Encoding.EncodingFromString() : Encoding.UTF8;

     bodies.Add(
                new MimePart
                        {
                            Content = mediaType ==
MediaTypeNames.Text.Html ?
encoding.GetBytes(HttpUtility.HtmlEncode(content.Content)) :
encoding.GetBytes(content.Content),
                            ContentType = mediaType
                        });
}

// add bodies
msg.Body = bodies.ToArray();

// add attachments
var attachments = new List<IncaMail.Attachment>();
attachments.Add(new Attachment
                    {
                       Name = attachment.Name,
                       Body = new MimePart
                       {
                          ContentType = mimeType,
                          Content = StreamHelper.GetBytes(zStream)
                       }
                     });
msg.Attachment = attachments.ToArray();
```

NOTE: If you send a HTML body text, use HttpUtility.HtmlEncode for special characters (e.g. ü for ü): http://dev.w3.org/html5/html-author/charref. Don´t use base64-encoding for the body content!

4.) Create IncaMailPortClient and send the message object

Send message with IncaMailPortClient SendMessage object	

C#	`var incaMailClient = new IncaMailPortClient();`
	
	`// set your IncaMail username and password`
	`incaMailClient.ClientCredentials.UserName.UserName = "UserName";`
	`incaMailClient.ClientCredentials.UserName.Password = "Password";`
	
	`using (new OperationContextScope(incaMailClient.InnerChannel))`
	`{`
	`  // Add a HTTP Header to an outgoing request`
	`  var requestMessage = new HttpRequestMessageProperty();`
	`requestMessage.Headers["Authorization"] = "Basic " + Convert.ToBase64String(Encoding.Default.GetBytes(incaMailClient.ClientCredentials.UserName.UserName + ":" + incaMailClient.ClientCredentials.UserName.Password));`
	
	`OperationContext.Current.OutgoingMessageProperties[HttpRequestMessageProperty.Name] = requestMessage;`
	
	`  // send your message`
	`  var response = incaMailClient.SendMessage(msg);`
	`}`

5. REST Send API

5.1 Goal

The REST API enables the functionality of IncaMail to be integrated in business and web applications.

This chapter addresses developers who want to add IncaMail functionality to their business solutions by using REST. They learn the scope of services, how to quickly run and test examples, and how to call IncaMail services from within their applications.

5.2 REST Send API

5.2.1 Requests

The IncaMail REST API contains the following requests:

API	*Description*
GET /ping	Ping for a connection check
POST /message/send	Send an IncaMail message
POST /message/prepareFile	Prepare a potentially large file (up to 1GB) to be used as a large file attachment in a subsequent call to "message/send"
POST /sentMessages/getDeliveryState	Get message states by MsgIDs

5.2.2 Authentication

Every REST Send API request uses Basic Auth to authenticate itself using an email address and a password. The email address has to be registered on IncaMail and a password has to be given. The password used for the IncaMail web application is the same as the one for the REST Send API.

Platform	*Web Application*
Integration Platform	https://sta.incamail.com
Production Platform	https://incamail.com

NOTE: Use the Integration Platform during the development of an IncaMail API integration. Switch to the Production Platform for productive usage!

5.2.3 Authorization

Every email address using the IncaMail REST Send API has to be authorized by the Swiss Post. Please contact the business support team at business@incamail.ch and list the sender email addresses to be used and the requested platforms.

> ***NOTE:*** The email addresses have to be registered on the IncaMail platform(s) before the authorization request is sent to the business support team by email.

5.2.4 OpenAPI Documentation

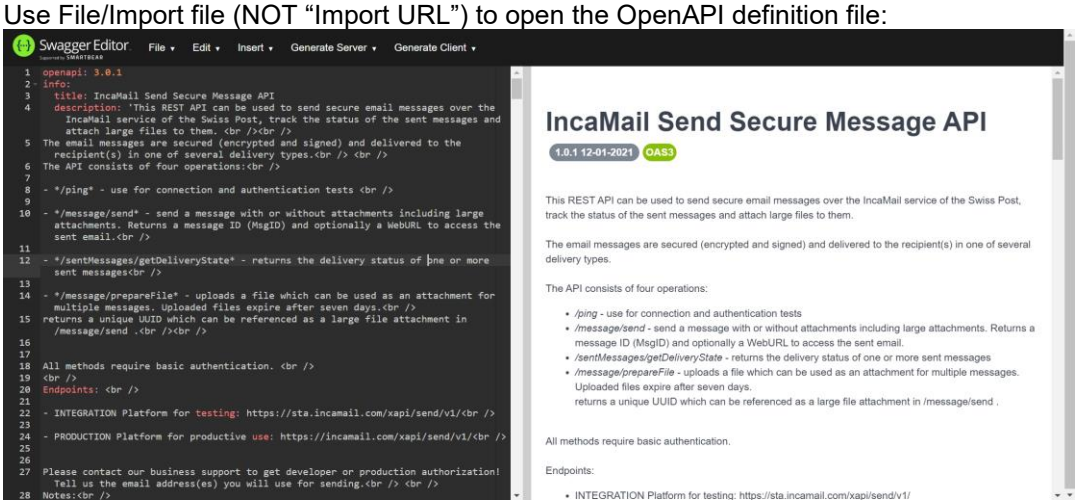
Online-description: https://incamail.com/apidef/restsend/index.html

The actual API documentation can be downloaded here (in an OpenAPI format):

https://incamail.com/apidef/restsend/im_rs_send.yaml

You can use your own REST developer tools to use it or use https://editor.swagger.io :

1. Download the YAML file from https://incamail.com/apidef/restsend/im_rs_send.yaml
2. Open https://editor.swagger.io
3. Use File/Import file (NOT “Import URL”) to open the OpenAPI definition file:



Editor.swagger.io lets you read the formatted documentation very easily and make test calls to the IncaMail API.

5.2.5 Getting Started: Explore the API With Postman

The following examples use the integration platform on https://sta.incamail.com/xapi/send/v1/ .

Requirements:

- Install the app Postman from https://postman.com/downloads/
- Email and password must be registered on the IncaMail integration platform https://sta.incamail.com
- The business support business@incamail.ch must have granted access rights to the REST Send API for the registered email address.
- Subject, body, attachments and other elements must be base64-encoded. The encoded strings in the examples are partly shortened and have to be replaced by complete ones.

REST Send API Postman Body Example: PING Connection Check	
/ping	`Leave the body empty!`
RESULT:	`200 OK`

REST Send API Postman Body Example: Send a simple text message	

/message/send	{
	"header": {
	"from": "sender@example.com",
	"to": [
	"recipient@example.com"
	],
	"subject": "VGhpcyBpcyBhIFJFU1QgdGVzdCBtZXNzYWdl"
	},
	"bodies": [
	{
	"content": "SGFsbG8=",
	"contentType": "text/plain"
	}
	],
	"props": {
	"deliveryType": "confidential"
	}
	}

RESULT:	{"msgID":"44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com"}

REST Send API Postman Body Example: Track the status of a message	
/sentMessages/ getDeliveryState	["44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com "]

RESULT:	[
	{
	"msgID": "44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com",
	"states": [
	{
	"state": "delivered",
	"recipient": "recipient@example.com",
	"stateChangedTime": "11/18/21 7:04:25 PM CET",
	"channel": "smtp",
	"safe": true
	}
	]
	}
	]

The following example creates an email with maximal protection:

- Delivery type “personal” (always encrypted, recipient must use an IncaMail password read the message)
- A text file attachment
- Personal, unencrypted message on the envelope. Builds up trust for the recipient.

- Encrypted subject

REST Send API Postman Body Example: Send a personal text message with an attachment	

/message/send	{
	"header": {
	"from": "sender@example.com",
	"to": [
	"recipient@example.com"
	],
	"subject": "VmVydHJhdWxpY2hlIFVudGVybGFnZW4",
	"safeEnvelopeMsg": "V2llIGdlc3Rlcm4gdGVsZWZvbmlzY2ggYmVzcHJvY2hlbiEKTEcgTWFydGlu"
	},
	"bodies": [
	{
	"content":
	"R3V0ZW4gVGFnCgpCZWlsaWVnZW5kIGZpbmRlbiBTaWUgZGllICp2ZXJ0cmF1bGljaGVuKiBVbnRlcmxhZ2VuLiBBdX
	MgRGF0ZW5zY2h1dHpncsO8bmRlbiBlcmhhbHRlbiBTaWUgZGllc2Ugw7xiZXIgSW5jYU1haWwuCgpGcmV1bmRsaWNoZ
	SBHcsO8c3NlClAuIEdlcmJlcg==",
	"contentType": "text/plain"
	}
	],
	"props": {
	"deliveryType": "personal",
	"hideSubject": "true"
	} ,
	"attachments": [
	{
	"mimeAttachment": {
	"contentType": "text/text",
	"content": "SGVsbG8gd29ybGQhIFRoaXMgaXMgYSB0aW55IGRvY3VtZW50Lg",
	"name": "Tiny little document.txt",
	"contentTransferEncoding": "base64",
	"contentInputEncoding": "",
	"contentDisposition": "attachment"
	}
	}
	]
	}

RESULT:	{"msgID":"44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com"}

Copyright Swiss Post

The following example creates a complex email:

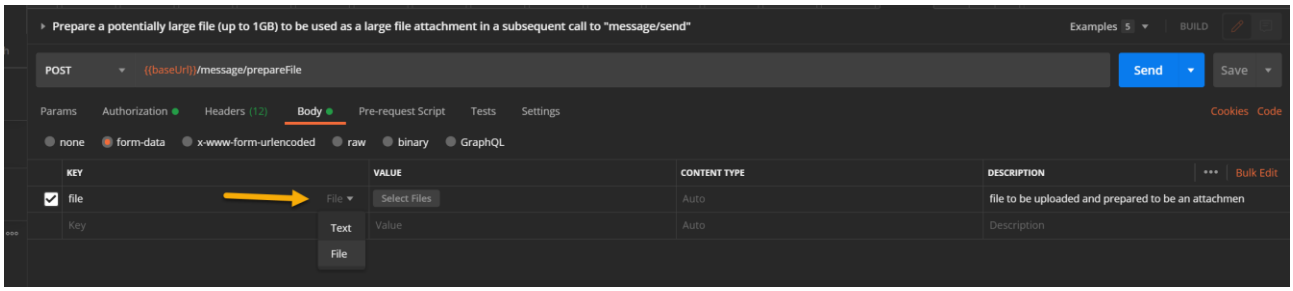
- Delivery type “plain_signed” (signed by the Swiss Post, but not encrypted)
- Two body parts for text and html display
- A text file attachment
- Inline image attachment which is embedded into the html body
- Large file attachment which is pre-loaded to IncaMail using `/message/prepareFile` before the message is sent. The preloaded file can be used for multiple messages. Large files are available seven days for download by a link embedded into the IncaMail email.

Preparation

1. Define the email parts and base64-encode them. Make sure the inline image has the correct name:

Element	***Content***	***Base64-encoded***
Subject	Vertrauliche Unterlagen	VmVydHJhdWxpY2hlIFVudGVybGFnZW4
SAFE envelope message	Wie gestern telefonisch besprochen! LG Martin	V2lllGdlc3Rlcm4gdGVsZWZvbmlzY2ggYmVzcHJvY2hl biEKTEcgTWFydGlu
HTML Body:	Guten Tag Beiliegend finden Sie die <i>vertraulichen</i> Unterlagen. Aus Datenschutzgründen erhalten Sie diese über IncaMail. <img width="66" height="93" style="width:66px;height:93px" src="cid:bono.jpg"> Freundliche Grüsse P. Gerber	PHA+R3V0ZW4gVGFnPC9wPgo8cD48L3A+CjxwPkJla WxpZWdlbmQgZmluZGVuIFNpZSBkaWUgPGk+dmVyd HJhdWxpY2hlbjwvaT4gVW50ZXJsYWdlbi4gQXVzIERh dGVuc2NodXR6Z3LDvG5kZW4gZXJoYWx0ZW4gU2lII GRpZXNlIMO8YmVyIDxiPkluY2FNYWlsPC9iPi48L3A+ CjxwPjwvcD4KPHA+PGltZyB3aWR0aD0iMjIzIiBoZWlna HQ9IjE2NiIgc3R5bGU9IndpZHRoOjY2cHg7aGVpZ2h0 OjkzcHgiIHNyYz0iY2lkOmJvbm8uanBnIj48L3A+CjxwPj wvcD4KPHA+RnJldW5kbGljaGUgR3LDvHNzZTwvcD4 KPHA+UC4gR2VyYmVyPC9wPg==
TEXT Body	Guten Tag Beiliegend finden Sie die *vertraulichen* Unterlagen. Aus Datenschutzgründen erhalten Sie diese über IncaMail. Freundliche Grüsse P. Gerber	R3V0ZW4gVGFnCgpCZWlsaWVnZW5kIGZpbmRlbiBT aWUgZGllICp2ZXJ0cmF1bGljaGVuKiBVbnRlcmxhZ2Vu LiBBdXMgRGF0ZW5zY2h1dHpncnS08bmRlbiBlcmhhbH RlbiBTaWUgZGllc2Ugw7xiZXIgSW5jYU1haWwuCgpGc mV1bmRsaWNoZSBHcsO8c3NlClAuIEdlcmJlcg==
MIME attachment: Text	Hello world! This is a tiny document.	SGVsbG8gd29ybGQhIFRoaXMgaXMgYSB0aW55IGRv Y3VtZW50Lg
MIME inline picture: bono.jpg		/9j/4AAQSkZJRgABAAEAkACQAAD//gAfTEVBRCBUZWNobm9sb2dpZXMgSW5jLiBWM S4wMQD/2wCEABkZGSgbKD8mJj8/LS0tP0Q8PDw8RERERERERERERERERERERERER ERERERERERERERERERERERERERERERERERERERERREQBGygoMiYyPCYmPEQ8MjxERE RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERER ERERERP/EAaIAAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKCwEAAwEBAQE BAQEBAQAAAAAAAAECAwQFBgcICQoLEAACAQMDAgQDBQUEBAAAAX0BAgMABB EFEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Nj c4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaX mJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy 8/T19vf4+foRAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBR CkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZX WFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7 i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/AABEIAF0AQgMBEQAC EQEDEQH/2gAMAwEAAhEDEQA/AM4uTx3qLG5ehicY+U1LTZSaR0QmWNRnr6VsczIjfL QApulkGP501oS9SNz8vHNTJ6FIzX3Nx0B9P61kUIg3oABwtAyrvlHGelAEUUihvkAJH8R/ pW1ibmsp8tdzHJoEUZbncaAK4lNMCcS0gLEVx260eoEpAbOPSsmrFIlSPYhU8g9u9IZX8 pP9qgZn6fGG+Y9BW7MkWbm4UDaOTQMzSc0xCigYbz0oEPRyvNAFwXH8Q6jtUNDLy SCVcrxmsmrFok2igDMhK28QDd+Tjqa2IKk3zfMAQD64/pVCIlPFACnpQBHTEAkIPFAFrK sM9D7GkMk0+bZJsPRv51MkNM39grEszLXa0YbritiBJBGTuHQZOCB+QPpTAzMHPApi EYHFMCLHFAgSItTAPunHcUgFhP79f94UnsPqdjisCzJgi8qMJ7VsSTGEBaQFEpk4pgSt COKAKdxFxx2qhFA0xBSARTiQH3BpPYOp041AAYIOe9ZWLGqM81oIhuboL8q84pAU1lJ 5NADxcLnmmA52Vwdpz7UCMsiqENI7UAQqcsP89KGCLHmP60hnQ79qe5FSMzFj8xvW kMdLGU4UUCKhBB560wEBwciqEOYcH86QFCRsn6VQgXgE/hQAYNIZ0U/Qj0qBkunqsY y3U5IzQBNKUZu2aYFC42LwOtAGTI5R8dxTQiYtkH3FAygwyTiqJLdooZwuM8jOe9Sxo6j dGP4E/wC+f/rVAyq+JUDrzn/P6VQFN2ORtOMfypDGSoA3Bz6UXAoNIA3AJI/KqEQEc5P WgRJngn8qBlcHmqEaOlLmfHoCaVrjR03lj1o5SrGBBcGHK9VPb/CkQEu1jlTj2qRkIPOAc0 DI3LdKYiIJk0AMlbBwOgqkIgPWgDX0hcyt7Lge5J6U0NG+WxWhZy5NYmYzcSKAI6YEgY ikA7JPWgCCQUxEPegZuaamFZu5bH5f/rpAbAuB/dBp3NLH/9k=

2. Upload the large file attachment using `/message/prepareFile`. This request requires form-data to be sent. In Postman, a file can be attached very simple:



Use “Select Files” to define a file to be uploaded.

The API will returen a JSON containing a UUID which is used to reference the uploaded file later:

```
{
    "UUID": "MekeP6aTsVlwfdnhqofKFP7LXqzoZ2Dp",
    "originalfilename": "im_rs_send.yaml.txt",
    "encoding": "UTF-8",
    "mimetype": "text/plain",
    "expires": "Oct 27, 2021 6:33:54 PM",
    "hash":
"04afd12141f56fed6d8040370c0448ad0cb2fbd37d27498716c03582577a3050",
    "size": 24758
}
```

3. Send the message:

REST Send API Postman Body Example: Send a plain, signed html message with an embedded image	

/message/send	{
	"header": {
	"from": "sender@example.com",
	"to": [
	"recipient@example.com"
	],
	"subject": "VmVydHJhdWxpY2hlIFVudGVybGFnZW4",
	"safeEnvelopeMsg": "TGllYmUgR3LDvHNzZSwgDQpNYXJ0aW4",
	"destinationAsyncMsg": "",
	"destinationReceipts": ""
	},
	"bodies": [
	{
	"content":
	"R3V0ZW4gVGFnDQoNCkJlaWxpZWdlbmQgZmluZGVuIFNpZSBkaWUgdmVydHJhdWxpY2hlbiBVbnRlcmxhZ2VuLiBBdX
	MgRGF0ZW5zY2h1dHpncsO8bmRlbiBlcmhhbHRlbiBTaWUgZGllc2Ugw7xiZXIgSW5jYU1haWwuDQoNCkZyZXVuZGxpY
	2hlIEdyw7xzc2UNClAuIEdlcmJlcg",
	"contentType": "text/plain"
	},
	{
	"content":
	"PHA+R3V0ZW4gVGFnPC9wPgo8cD48L3A+CjxwPkJlaWxpZWdlbmQgZmluZGVuIFNpZSBkaWUgPGk+dmVydHJhdWxpY2
	hlbjwvaT4gVW50ZXJsYWdlbi4gQXVzIERhdGVuc2NodXR6Z3LDvG5kZW4gZXJoYWx0ZW4gU2llIGRpZXNlIMO8YmVyI
	DxiPkluY2FNYWlsPC9iPi48L3A+CjxwPjwvcD4KPHA+PGltZyB3aWR0aD0iNjYiIGhlaWdodD0iOTMiIHN0eWxlPSJ3
	aWR0aDo2NnB4O2hlaWdodDo5M3B4IiBzcmM9ImNpZDpib25vLmpwZyI+PC9wPgo8cD48L3A+CjxwPkZyZXVuZGxpY2h
	lIEdyw7xzc2U8L3A+CjxwPlAuIEdlcmJlcjwvcD4=",
	"contentType": "text/html"
	}
	],
	"props": {
	"deliveryType": "plain_signed"
	},
	"attachments": [
	{
	"largeFileAttachment": {
	"uuid": "MekeP6aTsVlwfdnhqofKFP7LXqzoZ2Dp"
	}
	},
	{

REST Send API Postman Body Example: Send a plain, signed html message with an embedded image


```
                "mimeAttachment": {
                    "contentType": "text/text",
                    "content": "SGVsbG8gd29ybGQhIFRoaXMgaXMgYSB0aW55IGRvY3VtZW50Lg",
                    "name": "Tiny little document.txt",
                    "contentTransferEncoding": "base64",
                    "contentInputEncoding": "",
                    "contentDisposition": "attachment"
                }
            },
            {
                "mimeAttachment": {
                    "contentType": "image/png",
                    "name": "bono.jpg",
                    "contentTransferEncoding": "base64",
                    "contentInputEncoding": "base64",
                    "contentDisposition": "inline",
                    "content":
"/9j/4AAQSkZJRgABAAEAkACQAAD//gAfTEVBRCBUZWNobm9sb2dpZXMgSW5jLiBWMS4wMQD/2wCEABkZGSgbKD8mJj
8/LS0tP0Q8PDw8RERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERQBGygoMiYyPCYmPEQ8M
jxERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERERP/EAaIAAAEFAQEBAQEBAAAA
AAAAAAABAgMEBQYHCAkKCwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoLEAACAQMDAgQDBQUEBAAAAX0BAgMABBE
FEiExQQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2
RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4
eLj5OXm5+jp6vHy8/T19vf4+foRAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1Lw
FWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipK
TlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/AABEIAF
0AQgMBEQACEQEDEQH/2gAMAwEAAhEDEQA/AM4uTx3qLG5ehicY+U1LTZSaR0QmWNRnr6VsczIjfLQApulkGP501oS9S
Nz8vHNTJ6FIzX3Nx0B9P61kUIg3oABwtAyrvlHGelAEUUihvkAJH8R/pW1ibmsp8tdzHJoEUZbncaAK4lNMCcS0gLEV
x260eoEpAbOPSsmrFIlSPYhU8g9u9IZX8pP9qgZn6fGG+Y9BW7MkWbm4UDaOTQMzSc0xCigYbz0oEPRyvNAFwXH8Q6j
tUNDLySCVcrxmsmrFok2igDMhK28QDd+Tjqa2IKk3zfMAQD64/pVCIlPFACnpQBHTEAkIPFAFrKsM9D7GkMk0+bZJsP
Rv51MkNM39grEszLXa0YbritiBJBGTuHQZOCB+QPpTAzMHPApiEYHFMCLHFAgSItTAPunHcUgFhP79f94UnsPqdjisC
zJgi8qMJ7VsSTGEBaQFEpk4pgStCOKAKdxFxx2qhFA0xBSARTiQH3BpPYOp041AAYIOe9ZWLGqM81oIhuboL8q84pAU
1lJ5NADxcLnmmA52Vwdpz7UCMsiqENI7UAQqcsP89KGCLHmP60hnQ79qe5FSMzFj8xvWkMdLGU4UUCKhBB560wEBwci
qEOYcH86QFCRsn6VQgXgE/hQAYNIZ0U/Qj0qBkunqsYy3U5IzQBNKUZu2aYFC42LwOtAGTI5R8dxTQiYtkH3FAygwyT
iqJLdooZwuM8jOe9Sxo6jdGP4E/wC+f/rVAyq+JUDrzn/P6VQFN2ORtOMfypDGSoA3Bz6UXAoNIA3AJI/KqEQEc5PWg
RJngn8qBlcHmqEaOlLmfHoCaVrjR03lj1o5SrFl5Srh1/CkQEFdNO4A2WNIzdZMhoNoD6/MQVR5D3x/qEO3VVjwd3q/
BhDr7jOOmtmdS03VlwWkxDcC9Wkg+2XPLbcHyXYumvJVjEMLzWTAlvDwh4YS3zpvVB3KOsaiN1G6RbHs1DmQvaw7wKp2
RJngn8qBlcHmqEaOlLmfHoCaVrjR03lj1o5SrFl5Srh1/CkQEFdNO4A2WNIzdZMhoNoD6/MQVR5D3x/qEO3VVjwd3q/
IgPWgDX0hcyt7Lge5J6U0NG+WxZWZm5NYlLwP5T+bVfLOkzmnfAyO0NbaMZPZcTB/rpAbAuB/dBp3NL
H/9k="
                }
            }

    ]
}
```

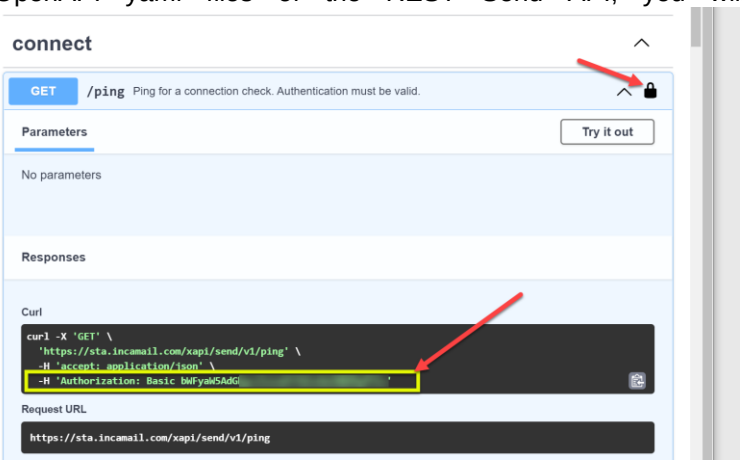

RESULT: {"msgID":"44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com"}

5.2.6 Examples in Excel

The following examples use the integration platform on https://sta.incamail.com/xapi/send/v1/ .

Requirements:

- Email and password must be registered on the IncaMail integration platform https://sta.incamail.com
- The business support business@incamail.ch must have granted access rights to the REST Send API for the registered email address.
- Subject, body, attachments and other elements must be base64-encoded. The encoding itself is not done in the examples.
- The authorization string for the username/password must be included in the header of the request. Use a tool or an Excel macro to generate the correct string. If you use editor.swagger.io to open the OpenAPI yaml files of the REST Send API, you will get the access string directly:



REST Send API Excel Example: PING Connection Check	

/ping	`Sub Ping()`
	`  Dim objRequest As Object`
	`  Dim strURL As String`
	`  Dim blnAsync As Boolean`
	`  Dim strResponse As String`
	`  Dim strStatus As String`
	`  Dim strAuth As String`
	
	`  strAuth = "bWFbseZdekZseaJ3ZHhd2348Hdfs8KEx"`
	
	`  Set objRequest = CreateObject("MSXML2.XMLHTTP")`
	`  strURL = "https://sta.incamail.com/xapi/send/v1/ping"`
	`  blnAsync = True`
	
	`  With objRequest`
	`  .Open "GET", strURL, blnAsync`
	`  Rem .SetRequestHeader "Content-Type", "application/json"`
	`  .setRequestHeader "accept", "application/json"`
	`  .setRequestHeader "Authorization", "Basic " & strAuth`
	`  .Send`
	`  While objRequest.readyState <> 4`
	`  DoEvents`
	`  Wend`
	`  MsgBox .Status & "  " & .statusText & vbNewLine & .responseText`
	`  End With`
	`End Sub`

REST Send API Excel Example: Send a simple text message	
/message/send	

```
Sub Send()
    Dim objRequest As Object
    Dim strStatus As String
    Dim strAuth As String
    Dim strURL As String
    Dim strFrom As String
    Dim strTo As String
    Dim strJSON As String

    strFrom = "sender@example.com"
    strTo = "recipient@example.com"
    strAuth = "bWFbseZdekZseaJ3ZHhd2348Hdfs8KEx"

    strJSON = "{"
    strJSON = strJSON & """header"": {"
    strJSON = strJSON & """from"": """ & strFrom & ""","
    strJSON = strJSON & """to"": [""" & strTo & """],"
    strJSON = strJSON & """subject"": ""VGhpcyBpcyBhIFJFU1QgdGVzdCBtZXNzYWdl"""
    strJSON = strJSON & "},"
    strJSON = strJSON & """bodies"": [{""content"": ""SGFsbG8="",""
    strJSON = strJSON & """contentType"": ""text/plain""}],"
    strJSON = strJSON & """props"": {""deliveryType"": ""confidential""}"
    strJSON = strJSON & "}"

    Set objRequest = CreateObject("MSXML2.ServerXMLHTTP")
    strURL = "https://sta.incamail.com/xapi/send/v1/message/send"

    With objRequest
        .Open "POST", strURL, True
        .setRequestHeader "Content-type", "application/json"
        .setRequestHeader "accept", "application/json"
        .setRequestHeader "Authorization", "Basic " & strAuth
        .Send (strJSON)

        While .readyState <> 4
            DoEvents
        Wend
        strStatus = .Status & "  " & .statusText & vbNewLine & .responseText
        MsgBox strStatus

    End With
End Sub
```

RESULT:

```
{"msgID":"44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com"}
```

REST Send API Excel Example: Track the status of a message	
/sentMessages/ getDeliveryState	

```
Sub Status()
    Dim objRequest As Object
    Dim strStatus As String
    Dim strAuth As String
    Dim strURL As String
    Dim strJSON As String

    strAuth = "bWFbseZdekZseaJ3ZHhd2348Hdfs8KEx"
    strJSON = "[""44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com""]"

    Set objRequest = CreateObject("MSXML2.ServerXMLHTTP")
    strURL = "https://sta.incamail.com/xapi/send/v1/sentMessages/getDeliveryState"

    With objRequest
        .Open "POST", strURL, True
        .setRequestHeader "Content-Type", "application/json"
        .setRequestHeader "accept", "*/*"
        .setRequestHeader "Authorization", "Basic " & strAuth
        .Send (strJSON)

        While .readyState <> 4
            DoEvents
        Wend
        strStatus = .Status & "  " & .statusText & vbNewLine & .responseText
        MsgBox strStatus
    End With
End Sub
```

RESULT:

```
[
    {
        "msgID": "44fe75ea-8544-4b25-8002-4483ec839a71@sta.incamail.com",
        "states": [
            {
                "state": "delivered",
                "recipient": "recipient@example.com",
                "stateChangedTime": "11/18/21 7:04:25 PM CET",
                "channel": "smtp",
                "safe": true
            }
        ]
    }
]
```