IndusSoft Technologies

# Email Tunnel
# Software Architecture Document

**Version 1.0**

## Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| 11/24/2000 | 1.0 | Software architecture document for Email Tunnel | Imran Hussain |
| | | | |
| | | | |
| | | | |

# Table of Contents

# Software Architecture Document

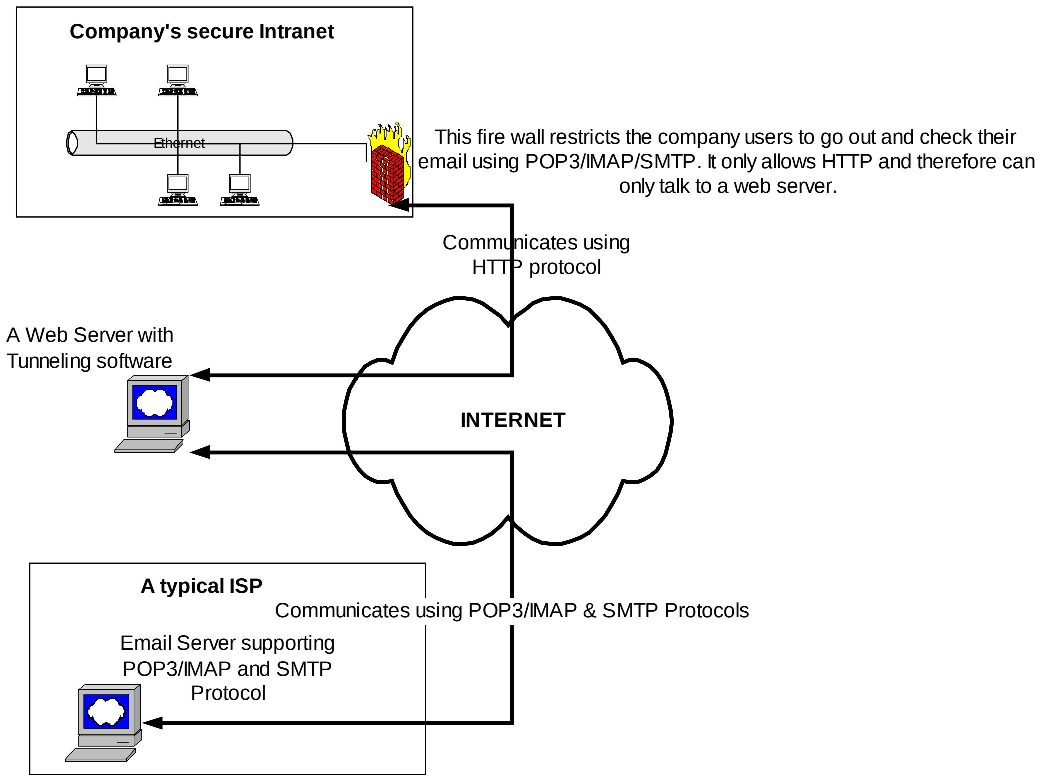
## 1. Introduction

Email Tunneling is a mechanism to check personal emails using HTTP protocol as supposed to POP3/IMAP. This document describes the design of this system. Since there is no separate Vision document for this system, the beginning part of this matter also serves as a vision document, which specifies high-level requirements and design constraints. Besides, it provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

### 1.1 Requirements overview

Many organizations, for security reasons, allow restricted network traffic through their corporate firewalls. In fact, many companies allow only Web access to the outside world limiting the traffic to HTTP and HTTPS protocols. This poses a problem for people who want to check secondary emails where the email server resides outside the intranet.

Email Tunneling allows POP3/IMAP/SMTP protocols to be converted to HTTP so that users can check their emails behinds a firewall. Refer to the diagram below for a pictorial description.


Company's secure Intranet
Ethernet
This fire wall restricts the company users to go out and check their email using POP3/IMAP/SMTP. It only allows HTTP and therefore can only talk to a web server.
Communicates using HTTP protocol
A Web Server with Tunneling software
INTERNET
A typical ISP
Communicates using POP3/IMAP & SMTP Protocols
Email Server supporting POP3/IMAP and SMTP Protocol

Here a web server acts as a proxy for email and gets the required data from a remote server which is then sent to the client as plain HTML.

## 1.2 Definitions, Acronyms and Abbreviations

- **HTTP:** Hyper text transport protocol - Used by web browsers to communicate with a web server
- **HTTPS:** Hyper text transfer protocol secure - Used by web browsers to communicate with a web server on a secured channel
- **SMTP:** Simple mail transfer protocol - Used by email clients like Netscape messenger, MS Outlook to send an email message
- **IMAP:** Internet message access protocol - Used by email clients to read email message in their INBOX
- **POP3:** Post office protocol 3 - Used by email clients to read email messages in their INBOX
- **JAXP:** Java API for XML Processing - Used to read and store email profiles in the system.

## 1.3 References

This system uses a bunch of technologies from Sun Microsystems. You can read about them in detail at the following sites:

- **JDK 1.3:** Java SDK 1.3 - http://java.sun.com/j2se/1.3/
- **Java Servlets:** Servlets - http://java.sun.com/products/servlet/index.html
- **Java Mail:** Java Email Specification - http://java.sun.com/products/javamail/index.html
- **JAXP:** Java API for XML Processing - http://java.sun.com/xml

## 1.4 Technology Overview

Email Tunnel is a part of software that runs on a web server. It uses Java Servlet technology to communicate between an email server that supports either POP3 or IMAP protocol for reading email messages and SMTP for sending. All the mail specific communication is handled by Java Mail, a set of API provided by Sun Microsystems that model a mail system. It also uses JAXP for storing and retrieving email profiles in an XML file.

## 2. Logical View

Logically, the system is divided into three groups:

- Presentation
- Logic controller
- Communication with remote mail server

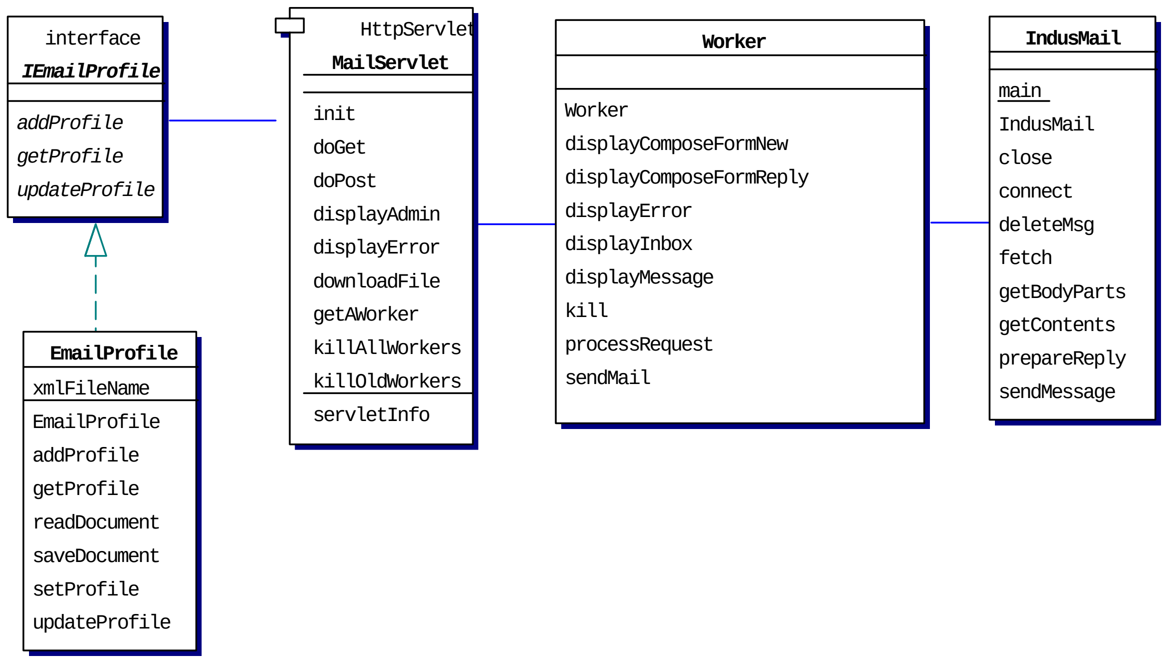
The presentation layer is a based on static HTML templates that are read by the servlets and the result is sent back to the client. The reason for these templates is to avoid code compilation if some cosmetic change is made to the site.

The controller layer parses all the requests that come in from the browser and redirects it to appropriate handler. These handlers then either call the communication layer or simply use the static HTML pages to send a message.

Finally, the communication layer is responsible for all the mail specific messages between the web server and email server.

### 2.1 Class diagram

Following class diagram depicts the physical design of the system.



This is a logical view of the classes. Actual class has other methods both private and public that are not shown in the above picture.

### *2.1.1 MailServlet*

MailServlet class implements HTTPServlet and therefore, its objects are called by the web server. It implements doGet and doPost methods that are invoked by the servlet engine. It also maintains a list of Worker objects that creates workers when a new client comes in and deletes them when they are no longer needed.

This servlet uses two parameters to perform its tasks. First is a session id that is used for state management and the other is operation id that is used to specify the task user wants to perform. Both of these parameters are passed back from the client using URL redirection. URL redirection eliminates the use of cookies on the client browser, which is an alternate approach to for state management. Internally, the class maintains a table that contains a mapping between session id and worker objects. A new worker is created if the session id is not present in the URL.

### *2.1.2 Worker*

The MailServlet object delineates all the work to a Worker object that is specific to user sessions. They contain data like user profile and has methods to display email contents, show inbox listing, and send email messages. This object inspects the operation variable and depending on its value calls different methods within itself. For scalability reasons, the worker object is separated from the servlet. If later down the road we need to distribute some load to an application server, we can easily move the worker objects in an EJB container.

### *2.1.3 IndusMail*

This wrapper class simplifies the calls exposed by Java Mail at higher level. It is responsible for reading message in an INBOX, displaying message contents with multiple parts and sending messages.

### *2.1.4 EmailProfile*

The system allows a user to save his/her email profile so that they do not have to enter all the server information every time they use the service. This information is stored in an XML file on the system. The reason for using an XML file verses a relational database is to avoid the overhead of a database and making the system more portable. Currently, the number of users who will save their profile is expected to be very small. If this number increases, an XML parser will not be able to handle the load and therefore will have to be moved to a database.

## 3. Deployment Requirements

Following is list of required libraries that you need to run the servlet.

- JSDK 2.0 - jsdk.jar
- Java Mail - mail.jar, pop3.jar
- JAXP 1.0 - jaxp.jar, parser.jar
- JDK 1.2 or above.

All the above software can be downloaded from http://www.javasoft.com

### 3.1 Configuration

Following entries must be added in the servlet engine for initial arguments:

- **IncFilePath:** Absolute path specifying the location for all the static HTML templates.
- **AttachmentPath:** Absolute path used to store email attachments. The worker object uses this directory to temporarily store files. When the kill method of the object is called, these directories are deleted. However, in case of unexpected shutdown, you might have to delete them manually.
- **ImagePath:** Absolute path where images are stored.
- **ProfilePath:** Name of the XML file to store profiles. Must have an absolute path.

## 4. Obtaining binaries and source code

If you wish to use this servlet on your web site, you may do that at no cost. You DO NOT have to pay any royalties. You can also obtain the source code for $199.00. Contact the author if you are interested in obtaining the source code for this servlet. The author's email address is imranh@imranweb.com