Integrated Message Framework: Strategy, Design and Implementation

by

Tarek Dajani

Bachelor of Architecture
American University of Beirut
1998

Submitted to the Department of Civil and Environmental Engineering
In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Civil and Environmental Engineering

At the

Massachusetts Institute of Technology

June 2004

©2004 Tarek Dajani. All rights reserved

The author hereby grants to MIT permission to reproduce and to distribute publicly paper
and electronic copies of this thesis documents in whole or in part.

Signature of author

Department of Civil and Environmental Engineering
May 07, 2004

Certified by

Dr. George Kocur
Senior Lecturer,
Department of Civil and Environmental Engineering
Thesis supervisor

Accepted by

Heidi Nepf
Chairman, Departmental Committee on Graduate Studies

BARKER
Integrated Message Framework: Strategy, Design and Implementation

By Tarek Dajani

Submitted to the Department of Civil and Environmental Engineering, May 7, 2004, in partial fulfillment of the requirements for the Degree of Master of Engineering in Civil and Environmental Engineering

Abstract

As we subscribe to more and more communication channels our ‘world of messages’ is becoming a complex matrix of dispersed information. Emailing, Blogging, Instant Messaging... have become customary daily activities for many.

While the wealth of media enriches our interaction experiences, it nonetheless constrains our ability to assimilate information from one location (information is stored online, on a local machine, and on the local network) and in one format.

As a matter of fact, given our reliance on multiple communication channels, we have to resort to using several distributed applications that lack a unified visual interface and storage structure. Result:

1- Incompatible message description and storage structures.
2- Incompatible message transfer and notification standards. As in the case of Email and Instant Messaging which despite their many similarities are still highly non interoperable.
3- No horizontal visibility across and within message media formats (Email, Blog, IM, etc...) preventing unified searches, correspondence logging and centralized information management.
4- Distributed contact and address repository, preventing access to a unified comprehensive ‘Address Book’.

Large corporations have the ability to resolve the issues above by forcing a top down standardization policy that regulates correspondence forms and formats between and across their business units. While that approach might prove fairly successful in homogeneous environments, it fails to address the needs of heterogeneous project workgroups which have neither the infrastructure nor the logistics to enforce a unifying framework.

This document presents a new approach towards the consolidation of multiple messaging structures (Emails, Blogs, Instant Message Transcripts, File Transfers/Sharing, Calendar Tasks and Events as well as their associated contacts and addresses) into an integrated framework that is accessible via subscriber services within projects. Furthermore, it will explore the possibilities of leveraging individual media properties (such as the concept of Presence in instant messaging) across the entire messaging framework as well as the introduction of intelligent agents that streamline message delivery through prioritized dispatching, scheduled delivery and error, event and task notification.

The thesis will cover the implementation of an Intelligent Project Management Client (ILINK) as a working example of the integrated messaging framework (IMF) through the
combined use of desktop components, web services, xml transmission schemas, and database storage.
The document concludes with a retrospective analysis and an outline of milestones for the extension of the integrated message framework (IMF).

Thesis supervisor: Dr. George Kocur
Title: Senior Lecturer, Department of Civil and Environmental Engineering
# Table of Contents

## CHAPTER I INTRODUCTION .............................................................................................................. 7
1. THESIS RATIONALE .......................................................................................................................... 7
2. ANTICIPATED OUTCOMES ................................................................................................................ 9
3. DOCUMENT ORGANIZATION .......................................................................................................... 10

## CHAPTER II EXISTING MESSAGING PRODUCTS AND STANDARDS ................................................. 12
1. OVERVIEW .................................................................................................................................. 12
2. MESSAGING PRODUCTS .................................................................................................................. 12
3. MESSAGING MARKUPS ................................................................................................................... 18
4. COMMON MESSAGE DENOMINATORS: PRELUDE TO AN INTEGRATED FRAMEWORK .............. 27

## CHAPTER III INTEGRATED MESSAGE FRAMEWORK (IMF) ............................................................... 37
1. WHAT IS IMF? ................................................................................................................................. 37
2. IMF OBJECTIVES ............................................................................................................................ 37
3. IMF ENVIRONMENT ......................................................................................................................... 38
4. IMF BUILDING BLOCKS .................................................................................................................... 39
5. INTEGRATING MESSAGE MEDIA .................................................................................................... 41
6. LEVERAGING THE MESSAGING STRUCTURE: INTELLIGENT INTEGRATED FRAMEWORK ........ 42

## CHAPTER IV IMF ARCHITECTURE .................................................................................................... 51
1. OVERVIEW .................................................................................................................................. 51
2. IMF DATABASE PROCEDURES ......................................................................................................... 51
3. IMF CORE CLASSES ......................................................................................................................... 52
4. IMF WEB SERVICES ........................................................................................................................ 68
5. IMF GENERIC MESSAGE SERVICES ............................................................................................... 69
6. IMF CLIENT ACCESS ........................................................................................................................ 71

## CHAPTER V I-LINK ARCHITECTURE ................................................................................................. 73
1. WHAT IS I-LINK? ............................................................................................................................. 73
2. I-LINK ENVIRONMENT AND DEPLOYMENT REQUIREMENTS ...................................................... 73
3. I-LINK VISUAL ARCHITECTURE ..................................................................................................... 74
4. I-LINK ARCHITECTURE ................................................................................................................... 77

## CHAPTER VI OUTCOMES, CHALLENGES AND FUTURE WORK ..................................................... 86
1. OVERVIEW .................................................................................................................................. 86
2. IMF – I-LINK SUMMARY ................................................................................................................ 86
3. DESIGN CHALLENGES .................................................................................................................... 86
4. DEVELOPMENT SHORTCOMINGS .................................................................................................. 89
5. FRAMEWORK EXPANSION POSSIBILITIES .................................................................................. 89

## BIBLIOGRAPHY ................................................................................................................................. 91

## APPENDIX A IMF CLASS PACKAGES ............................................................................................... 92

## APPENDIX B SELECTED ARCHITECTURE SEQUENCES ................................................................. 93
List of Figures
FIGURE II-1 RFC 2822 EMAIL STRUCTURE ............................................................... 20
FIGURE II-2 MSN IM TRANSCRIPT STRUCTURE ..................................................... 22
FIGURE II-3 XMPP PRESENCE STRUCTURE .......................................................... 23
FIGURE II-4 XMPP MESSAGE STRUCTURE .......................................................... 24
FIGURE II-5 RSS 2.0 STRUCTURE ......................................................................... 25
FIGURE II-6 VCAL EVENT STRUCTURE .................................................................. 26
FIGURE II-7 IMS MESSAGE PACKAGE CONSTITUENTS ....................................... 33
FIGURE II-8 IMS PROJECT .................................................................................... 34
FIGURE II-9 IMF MARKUP MAPPING .................................................................... 35
FIGURE II-10 IMS XML SAMPLE .......................................................................... 36
FIGURE III-1 IMF PRESENCE RELAY .................................................................... 40
FIGURE III-2 IMF PRESENCE INFORMED DISPATCH ......................................... 41
FIGURE III-3 FILE SHARING ................................................................................ 43
FIGURE III-4 FILE ATTACHMENT ......................................................................... 44
FIGURE III-5 FILE TRANSFER .............................................................................. 45
FIGURE IV-1 IMF 5 TIERS .................................................................................... 51
FIGURE IV-2 IMF SESSION .................................................................................. 52
FIGURE IV-3 IMF USER ........................................................................................ 52
FIGURE IV-4 IMF PROJECT .................................................................................. 53
FIGURE IV-5 IMF CONTACT ................................................................................ 53
FIGURE IV-6 IMF ADDRESS ................................................................................ 54
FIGURE IV-7 IMF ACCOUNT ................................................................................ 54
FIGURE IV-8 IMF MESSAGE HEADER .................................................................. 55
FIGURE IV-9 IMF MESSAGE ADDRESS ............................................................... 55
FIGURE IV-10 IMF MESSAGE CONTACT .............................................................. 55
FIGURE IV-11 IMF MESSAGE PACKAGE .............................................................. 56
FIGURE IV-12 IMF MESSAGE PART ..................................................................... 56
FIGURE IV-13 IMF NOTIFICATION PACKAGE ..................................................... 57
FIGURE IV-14 IMF PRESENCE ............................................................................. 57
FIGURE IV-15 IMF ROSTER .................................................................................. 57
FIGURE IV-16 IMF ACCOUNT MANAGER ............................................................ 59
FIGURE IV-17 IMF CONTACT MANAGER ........................................................... 60
FIGURE IV-18 IMF ADDRESS MANAGER ............................................................ 61
FIGURE IV-19 IMF MESSAGE MANAGER ............................................................ 62
FIGURE IV-20 IMF PRESENCE MANAGER ........................................................... 63
FIGURE IV-21 IMF USER MANAGER .................................................................. 64
FIGURE IV-22 IMF PROJECT MANAGER .............................................................. 65
FIGURE IV-23 IMF SESSION MANAGER ............................................................. 66
FIGURE IV-24 IMF WEB SERVICES ..................................................................... 69
FIGURE IV-25 PROJECT RSS FEED ..................................................................... 70
FIGURE IV-26 : IMF - I-LINK ARCHITECTURE .................................................... 72
FIGURE V-1 I-LINK MAIN INTERFACE ............................................................... 75
FIGURE V-2 I-LINK OBJECT-ACTION .................................................................. 76
FIGURE V-3 I-LINK CLIENT- IMF SERVICES COMMUNICATION .................................... 78
FIGURE V-4 IMF WEB SERVICE PROXY ............................................................ 78
FIGURE V-5 I-LINK WEB SERVICES COMMUNICATION .................................... 79
FIGURE V-6 LOGIN FORM .................................................................................... 80
FIGURE V-7 MAIN FORM .................................................................................... 82
FIGURE V-8 TCP NOTIFICATION ....................................................................... 85
Acknowledgments

I would like to start by thanking Pamela Chahine for her dedication and hard work throughout the past year. Most importantly, I would like to thank her for coping with me through the project’s ups and downs while being always a great friend. I would also like to express my gratitude to Professor Kocur, for his valuable insights, encouragements, and his relentless emphasis on quality.

Ali, Charles, and Randa, your support and faith have encouraged me to aim high and fight hard.

Wajih, I wouldn’t be here if it weren’t for you. I cannot thank you enough.

My Family:

Dana, our weekly phone calls were home, away from home. I love you and can’t wait till we’re all together in Beirut.

Zina, no words can express enough how precious your support and encouragements were in the past months. I love you so much, and promise to be there for you, always...

Mom, you always encouraged me to pursue the highest goals, and fueled my endeavors with love and caring...

Dad, you have given me the greatest gift of all: your ethics, wisdom, and most importantly your love. I will always live to make you proud...

Tarek
Chapter I  Introduction

1. Thesis Rationale

On average internet users rely on at least two communication media and in turn subscribe to several channels for each medium (Refer to: Jupiter Research, http://www.jupiterresearch.com). It is not uncommon for a casual Internet user to have several email addresses (e.g., one personal and one for work), along with subscriptions to multiple messaging clients (e.g., MSN Messenger, Yahoo). Furthermore, the recent rise in popularity of blogs has added yet another messaging channel.

The three media above have several variations, each with its own standards: email, Microsoft Messenger markup, Yahoo Messenger markup, XMPP, RSS etc..., creating even more diversity in message structure and presentation.

The need for an integrated messaging framework becomes obvious as we attempt to navigate through sets of messages using a different client for each medium, sometimes even multiple clients for the same medium (e.g., webmail and Outlook for email). Each client stores its data using a proprietary structure and presents it in a different format making it hard for a user to access and manipulate information from one visual interface. Furthermore, diverse storage locations and incompatible storage formats make it hard for users to access all the data at the same time. The need becomes even more pressing as we attempt to use these media within the context of project management and team collaboration since issues of contact management, correspondence logging, centralized document access and event and task management add yet another level of complication.

1.1. Software Project Management

The motivating factor behind software development ventures is a true, visible need. In our case, the need arises from our ongoing struggle as software developers working on various projects, to cope with diverse information flows from multiple sources. As a matter of fact, attempting to manage software projects can be quite challenging for development groups working in heterogeneous environments
where neither infrastructure nor group work processes are uniform. As such, teams struggle with issues such as:

- Managing projects across geographic locations
- Accessing multiple shared documents from within several clients.
- Communicating through several media such as instant messaging, blogs and email while maintaining correspondence logs.
- Sending project updates to external parties that do not use the same client or messaging medium (such as a senior project managers).
- Submitting scheduled document reviews.
- Insuring project progress and documents visibility for external users (such as customers) that use different project management and communication systems.

The challenges above are addressed by leveraging the features of the various messaging media, allowing centralized access to project information, while eliminating the user’s dependence on a specific delivery and dispatch channel/client.

The I-LINK Windows client uses the Integrated Message Framework’s capabilities to enable this collaboration.

1.2. Integrated Message Framework (IMF)

Integrated Message Framework is a set of subscriber services (web services) that enable diverse text messaging clients to have access to a common messaging and contact management platform.

IMF provides platform and client independent access to integrated text message repositories.

The framework allows clients to send, receive, retrieve, log and archive messages in various formats across multiple standard media such as email, instant messaging and blogs. It also provides common contacts and addresses structure that transcends media discrepancies.

The framework enables users to have access to their information regardless of media and clients, therefore allowing them to collaborate independent of
presentation and storage formats. The IMF repository can be accessed from blog aggregators through RSS feeds, email clients through mail notification, web and windows applications through web services consumption.

The framework introduces projects as base units for sharing and organizing messages, contacts and media accounts.

1.3. Intelligent Client (I-LINK)

The I-LINK client provides a project management environment that allows accessing, sharing and organizing messages and contacts through project units. It features a concise, user-friendly desktop interface which enables ‘one click’ access to information.

I-LINK provides development teams with a versatile project management and collaboration tool that allows them to manage their work centrally and deliver it across multiple media such as email, blog and instant messaging.

I-LINK is a .Net Windows application that derives its main functionality from the IMF Services, and uses the IMF Database as a central information repository. I-LINK acts as a thin IMF client that subscribes to IMF’s web services for its core messaging functionality while handling message and event presentation within the Windows application.

2. Anticipated Outcomes

The project’s deliverables are:

a- An integrated message storage and transmission architecture that is used for email, IM, blog, file transfer/sharing, and calendar tasks and events, as well as a wrapper schema that allows the superimposition of intelligent logic onto existing messages, such as scheduled delivery, prioritized dispatching and error, task and event notifications.

b- An Integrated Message Framework that uses this architecture to enable unified message dispatch, delivery, notification, logging and storage through subscriber services (web services).
Intelligent project management client (I-LINK) that is built on the Integrated Message Framework, and allows software development groups to collaborate across heterogeneous environments. The system consists of a client application that consumes core services from a central subscription server.

3. Document Organization
The document is organized into six chapters.
Chapter (2) provides an analysis of the capabilities and limitations of existing messaging and small project management systems as well as an overview of current attempts to tackle problems such as message markup and transmission standards, centralized document repositories, and decentralized project management. It starts by exploring various messaging clients such as blog aggregators, email clients and online project managers.
It then provides an overview of existing messaging standards such as the RFC 2822 email standard, and the RSS 2.0 blog markup standard, presents an account of existing message formats and provides a detailed dissection of message structures (composition, section separation, different components, etc...) as well as an analysis of current message transmission architectures.
Finally it explores suggested standards such as XMPP (Extensible Messaging and Presence Protocol) and contact and calendar sharing standards.

Chapter (3) introduces the components that make up the Integrated Message Framework.
The chapter uses the previous analysis to derive common denominators between various message types which leads to identifying the IMF building blocks: Messages and Contacts.
It then introduces new concepts derived from individual media capabilities: user and media presence, message rules (delayed send, prioritized dispatching), error, event and task notification, etc... and identifies the strongest of these candidates for inclusion in the integrated framework and the project management client.
Chapter (4) provides an overview of the multiple services which will be implemented within the integrated message framework. It goes on to detail the definition of messages and contacts, provides an overview of the IMF types model as well as describes the various IMF architectural components. It also details the architectural components’ functionality, the way this functionality is exposed, its message storage and notification architecture, as well as the hardware and software infrastructures required.

Chapter (5) details the intelligent project management client (I-LINK) architecture, particularly in its relationship with the Integrated Message Framework, both on the level of exposed functionality and internal relationships.

Chapter (6) concludes the analysis through a retrospective outline of challenges faced as well as the identification of areas for potential future works.
Chapter II  Existing Messaging Products and Standards

1. Overview
In order to make sense of the existing Internet messaging landscape, we decided to analyze a comprehensive sample of messaging clients, as well as explore the various media standards that we use in most Internet text communication.

2. Messaging Products

2.1. Email Clients

2.1.1. Webmail
Webmail is an online tool that enables POP3 and IMAP email users to access their message repository through a regular internet browser such as Internet Explorer. Most webmail applications allow basic communication functionality such as sending and receiving mail and storing basic contact information (email address and display name). Such basic functionality, while quite useful for remote access to messages poses several challenges in terms of storage space, storage architecture and contact and message management (for users who own multiple email accounts).

While storage seems to be less and less of an issue with ever decreasing hardware costs (Google’s GMAIL featuring 1 GB of disk storage is a striking example), the storage architecture still presents significant challenges for integration within existing management applications and with other media formats.

Webmail, despite its interface ease of use and global access, suffers from lack of integration of multiple accounts within a central management interface. A user has to login to multiple webmail systems in order to retrieve mail messages and contact information for his/her different accounts. An alternative would be to forward all messages to a central account which has proved to be as problematic in that it does not allow a contextual separation of messages and creates an increasing bandwidth consumption across the network.
2.1.2. Desktop Mail Clients:
Email clients like Microsoft Outlook and Eudora are the most common email messaging applications. They allow basic communication such as sending and receiving mail as well as more complex tasks such as aggregating messages from multiple accounts and searching through local data stores. They provide detailed contact management features, as well as integrated task and event scheduling features.
Overall desktop based clients provide a more extensive feature list than web based systems. Nevertheless, desktop clients rely on local message storage and therefore cannot be accessed remotely as in the case of webmail (Protocols such as IMAP4 resolve that issue yet apply only to individual accounts and do not support multiple account aggregation).
In addition, clients like Microsoft Outlook organize their information in proprietary binary formats which make message access from external applications extremely difficult.

2.2. Blog Aggregators
(For a comprehensive list of blog aggregators visit: http://blogs.law.harvard.edu/tech/directory/5/aggregators)
The activity known as ‘blogging’ is not new to the internet. As a matter of fact, it originated with news groups and guestbook posts which allowed internet users to share thoughts and messages with each others in public forums. The creation of a syndication standard (RSS: Really Simple Syndication) consecrated blogging as a ‘formal’ internet communication format.
In simple terms, blogging allows users of blog aggregators to gather message posts from several sources (channels) into a centralized reading pane.

2.2.1. Web Hosted Aggregators:
Web Hosted aggregators are the equivalent of Webmail applications for blogs. As a matter of fact, these applications allow users to subscribe to their desired blog channels through an online interface and view their blog messages from
any web based client. As in the case of all web based applications, the core advantage is in the ability to access the message repository remotely without the need for additional software. The disadvantage of online aggregators is that most applications allow aggregating posts into their central database, yet from that point onwards rely on proprietary data structures for message storage. As such, while users can aggregate and view messages from multiple channels, they cannot re-aggregate these messages from the central repository through other aggregators. Furthermore, online aggregators suffer from the slow performance associated with the Internet, as well as a relatively limited message control features.

2.2.2. Desktop Aggregators:
Desktop blog aggregators are becoming more and more versatile as the ‘industry’ progresses. As a matter of fact there are now over 20 products available in the market, each featuring novel functionalities ranging from integration with Outlook and Internet Explorer to highly visual message notification and organization.
These clients, as in the case of most desktop applications, store information in proprietary format locally thus making message access remotely or from other local applications very difficult.

2.3. Instant Messaging
Instant messaging, despite being around for several years, was redefined recently as a ‘cutting edge’ approach to personal and corporate messaging. As a matter of fact, all major technology players have so far deployed one or another form of messaging clients (MSN Messenger, YAHOO Messenger, Flash Central, etc..) , the open source community is working on its own breed of applications ( Jabber Messenger, XMPP protocol, etc..) and corporate messaging solutions are mushrooming across the industry ( Omnipod, Groove, Lotus Sametime, etc..).
While usage, target audiences and architectures differ, there seems to be a consensus on the potential inherent in leveraging the concept of PRESENCE or STATE in order to deliver a new breed of messaging and collaboration platforms.

### 2.3.1. Proprietary Standard Messengers

Messenger clients like Microsoft MSN Messenger and YAHOO Messenger rely on proprietary messaging standards in order to instantiate, deliver and parse chat conversations. Such an approach requires users to subscribe to a central system and download a messaging application that understands the proprietary markup.

Conversation logs are saved as individual local xml files through the messenger client, yet message transcripts cannot be accessed from a structured message repository either locally or remotely, therefore making it very hard for users to manage them.

### 2.3.2. XML based Messengers

XML based messengers such as Jabber based clients provide a flexible messaging platform which can be applied to both personal and corporate correspondence.

These messengers rely mainly on the XMPP (Extensible Messaging and Presence Protocol) for structuring, instantiating, parsing and delivering chat conversations.

Other than the significant advantage of being protocol independent, these messengers suffer from the same disadvantage of their ‘proprietary’ counterparts.

### 2.4. Document Sharing and Project Management

Online document sharing applications such as PHPProject (refer to: http://www.phprojekt.com), Project.net (refer to: http://www.project.net), and eProject (refer to: http://www.eproject.com) are becoming more and more widespread as hardware costs go down. These applications allow internet users to access online internet file repositories, manage file versions and exchange project related correspondence.
Online document sharing is often coupled with project management applications to provide an online comprehensive application with features ranging from file sharing, email messaging, to time card systems and task scheduling. The above functionality is ideal for software development teams that need to frequently share data and progress information across geographical locations. The major disadvantage of these applications is that they often present too wide an array of functions that clutter the new user’s desktop and therefore require a high learning curve. Furthermore, these applications are mostly developed using proprietary data structures that complicate integration with other software therefore making web access the sole application entry point.
<table>
<thead>
<tr>
<th>Clients</th>
<th>Main Advantages</th>
<th>Main Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMF</td>
<td>Instant Centralized Message, Contact and Project Access Access from all types of Clients: Web, Windows applications and Blog Aggregator Web Services architecture enables platform independence Information Organized and Shared through projects Project Information can be shared with non system users Search Across Media Types</td>
<td>Requires Large Central Storage Space Web Services rely on High Bandwidth Connections</td>
</tr>
<tr>
<td>Online Project Manager</td>
<td>Centralized Access to System Messages and Projects Access from Web Clients Information Organized and Shared through projects</td>
<td>Requires Large Central Storage Space Requires System Account Does not cater for multiple message media Cannot be accessed from non web clients</td>
</tr>
<tr>
<td>Online Document Storage System</td>
<td>Centralized File Message Access Access from Web Clients</td>
<td>Requires Large Central Storage Space</td>
</tr>
<tr>
<td>Web Mail</td>
<td>Online Email Message and Contact Access Access from Web Clients</td>
<td>Requires Large Central Storage Space Requires System Account Does not cater for multiple message media</td>
</tr>
<tr>
<td>Microsoft Outlook</td>
<td>Local Email Message and Contact Access Aggregate Multiple Email Accounts Information Organized in Local Folders Search through Message Contents</td>
<td>Requires Large Local Storage Space Does not cater for multiple message media [handles file sharing/posting, newsgroups, etc.]</td>
</tr>
<tr>
<td>MSN Messenger</td>
<td>Instant Messaging Instant File transfer</td>
<td>Does not cater for multiple message media No Access to centralized Conversation Logs Requires System Account</td>
</tr>
<tr>
<td>Blog Aggregator</td>
<td>Access to Message RSS Posts Aggregate multiple Blog Channels</td>
<td>Does not cater for multiple message media No Message and Contact Access</td>
</tr>
<tr>
<td>Online Blog</td>
<td>Access to Message RSS Posts Aggregate multiple Blog Channels Publish Messages to Blog</td>
<td>Does not cater for multiple message media No Message and Contact Access</td>
</tr>
</tbody>
</table>
### Table II-2: Messaging Clients Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>IMF</th>
<th>Online Project Manager</th>
<th>Online Document Storage System</th>
<th>Web Mail</th>
<th>Microsoft Outlook</th>
<th>MSN Messenger</th>
<th>Blog Aggregator</th>
<th>Online Blog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Across Message Formats</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aggregate Data from multiple accounts</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aggregate Data from multiple sources (Email, IM, Blog, Files)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Require System Account for Data Access</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Access Data from Multiple Clients</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access Data from Web Browser</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Access Data from Blog Aggregator</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Instant File Transfer</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Share Files</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Share Contacts</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Publish Projects as RSS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publish Messages as RSS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publish Contacts as RSS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publish Files as RSS</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send/Receive Email</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Post/Aggregate Blogs</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>IM Conversations</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features:</th>
<th>May be</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
</table>

### 3. Messaging Markups

While we can list hundreds, if not thousands of messaging applications currently in use, it is easy to narrow our survey of messaging markups to a handful of established...
standards. As a matter of fact, if we disregard non disclosed proprietary standards (such as Outlook storage, MSN Messenger, Yahoo and Lotus Sametime protocols) we can focus our analysis on 5 text messaging markups: RFC 2822 email standard, MSN IM conversation transcript, Jabber XMPP protocol, blog RSS 2.0, VCal calendar standard.

3.1. Analysis Methodology

In order to set the groundwork for a comparative analysis of the various media markups, we chose to use broad categories that can encapsulate all the variations within media field types. As such we will be using the following categorization scheme derived from the RFC2822 standard:

- **Origination Date Field**
  The origination date field identifies the date and time the message was originally sent.

- **Originator Fields**
  Originator fields are field that contain information pertaining to the initiator of the message, such as the ‘From’ field and ‘Reply-To’ field.

- **Destination Fields**
  Destination fields are fields that contain information pertaining to the intended recipient/s of the message, such as the ‘To’ field and the ‘CC’ field.

- **Trace Fields**
  Trace fields contain identification fields that are used by servers to track messages. (e.g., Time Stamps, Receipt dates)

- **Identification Fields**
  Identification fields provide unique message identification information such as session ID and message ID

- **Informational Fields**
  Informational fields provide optional descriptive fields that help recipients understand a message’s content. (e.g., Subject, Send Date)

- **Body Fields**
  Body fields contain the main message along with any additional message parts such as attachments.
3.2. Markups: RFC 2822 Email

(Refer to: http://www.cse.ohio-state.edu/cgi-bin/rfc/rfc2822.html)

Prior to the RFC2822 email standard many email systems had their own message formatting. The RFC standard specifies that each message should have two parts:

3.2.1. A header that contains message information

3.2.2. A body that contains the text of the message and any other additional media attachments

---

<table>
<thead>
<tr>
<th>Field</th>
<th>Min occur.</th>
<th>Max occur.</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origination Date Field</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Originator fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>from</td>
<td>1</td>
<td>1</td>
<td>From: mailbox-list</td>
</tr>
<tr>
<td>sender</td>
<td>0</td>
<td>1</td>
<td>Sender: mailbox</td>
</tr>
<tr>
<td>reply-to</td>
<td>0</td>
<td>1</td>
<td>Reply-To: address-list</td>
</tr>
<tr>
<td>Destination address fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>0</td>
<td>1</td>
<td>To: address-list</td>
</tr>
<tr>
<td>cc</td>
<td>0</td>
<td>1</td>
<td>Cc: address-list</td>
</tr>
<tr>
<td>bcc</td>
<td>0</td>
<td>1</td>
<td>Bcc: (address-list / [CFWS])</td>
</tr>
<tr>
<td>Identification fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>message-id</td>
<td>0</td>
<td>1</td>
<td>Message-ID: msg-id</td>
</tr>
<tr>
<td>references</td>
<td>0</td>
<td>1</td>
<td>References: 1*msg-id</td>
</tr>
<tr>
<td>Informational fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>0</td>
<td>1</td>
<td>Subject: unstructured</td>
</tr>
<tr>
<td>comments</td>
<td>0</td>
<td>U</td>
<td>Comments: unstructured</td>
</tr>
<tr>
<td>keywords</td>
<td>0</td>
<td>U</td>
<td>Keywords: phrase *(&quot;&quot;,&quot; phrase)</td>
</tr>
<tr>
<td>Trace fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>received</td>
<td>1</td>
<td>1</td>
<td>Received: name-val-list &quot;;&quot; date-time</td>
</tr>
<tr>
<td>Body fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIME-Version</td>
<td>0</td>
<td>U</td>
<td>MIME-Version: 1.0</td>
</tr>
<tr>
<td>Content-type</td>
<td>0</td>
<td>U</td>
<td>Content-Type: multipart/alternative; (text/plain,text/html,image/gif,image/jpg,...)</td>
</tr>
<tr>
<td>charset</td>
<td>0</td>
<td>U</td>
<td>charset=&quot;iso-8859-1&quot;</td>
</tr>
</tbody>
</table>
3.3. Markups: MSN IM Conversation Transcript

While Microsoft does not disclose the MSN Messenger communication protocol fully (a limited protocol draft was submitted to the IETF, yet it does not expose the full MSN functionality - [http://www.hypothetic.org/docs/msn/ietf_draft.txt](http://www.hypothetic.org/docs/msn/ietf_draft.txt)), we were able to analyze the locally stored xml chat transcripts generated by the desktop messenger client. The transcript begins with a 'LOG' identification that traces the starting point and ending points of the conversation through the 'FirstSessionID' and 'LastSessionID' fields. From that point onwards, the transcript is composed of a list of message threads between all conversation parties. The 'message' component can come in different types: message (i.e. regular chat text), invitation (an invitation for an audio or file transfer session) and an invitation response (response to an invitation indication completion, failure or initiation).
<table>
<thead>
<tr>
<th>Field</th>
<th>Min occur</th>
<th>Max occur</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td></td>
<td></td>
<td><code>&lt;Log LogonName=&quot;asoums@hotmail.com&quot; FirstSessionID=&quot;1&quot; LastSessionID=&quot;3&quot;&gt;</code></td>
</tr>
<tr>
<td><strong>Origination Date Field</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Originator fields</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogonName</td>
<td>1</td>
<td>1</td>
<td><code>&lt;Log LogonName=&quot;asoums@hotmail.com&quot; FirstSessionID=&quot;1&quot; LastSessionID=&quot;3&quot;&gt;</code></td>
</tr>
<tr>
<td>Trace fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FirstSessionID</td>
<td>1</td>
<td>1</td>
<td><code>&lt;Log LogonName=&quot;asoums@hotmail.com&quot; FirstSessionID=&quot;1&quot; LastSessionID=&quot;3&quot;&gt;</code></td>
</tr>
<tr>
<td>LastSessionID</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Originator fields (From)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogonName</td>
<td>1</td>
<td>1</td>
<td><code>&lt;From&gt;&lt;User LogonName=&quot;asoums@hotmail.com&quot; FriendlyName=&quot;@asoums-&quot;/&gt;&lt;/From&gt;</code></td>
</tr>
<tr>
<td>FriendlyName</td>
<td>1</td>
<td>1</td>
<td><code>&lt;From&gt;&lt;User LogonName=&quot;asoums@hotmail.com&quot; FriendlyName=&quot;@asoums-&quot;/&gt;&lt;/From&gt;</code></td>
</tr>
<tr>
<td>Destination address fields (To)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LogonName</td>
<td>1</td>
<td>U</td>
<td><code>&lt;To&gt;&lt;User LogonName=&quot;tarekdajani@hotmail.com&quot; FriendlyName=&quot;tdaj&quot;&gt;&lt;/To&gt;</code></td>
</tr>
<tr>
<td>FriendlyName</td>
<td>1</td>
<td>U</td>
<td><code>&lt;To&gt;&lt;User LogonName=&quot;tarekdajani@hotmail.com&quot; FriendlyName=&quot;tdaj&quot;&gt;&lt;/To&gt;</code></td>
</tr>
<tr>
<td>Identification fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DateTime</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
3.4. Markups: Jabber XMPP Protocol

(Refer to: http://www.jabber.org/ietf/draft-ietf-xmpp-core-23.html)

The XMPP protocol is designed to help the exchange of messages and presence information in close to real time. Information exchange is done through streaming XML elements.

While XMPP allows for a fully integrated messaging and presence architecture, it nonetheless does not require the use of both concepts simultaneously and as such we found it appropriate to analyze each component separately:

3.4.1. XMPP Presence

Presence is used in XMPP to express an entity’s current network availability (offline, online as well as user defined sub states such as Away and Busy). Presence ‘stanzas’ are used to communicate status information between various network entities such as users/contacts and messaging servers.

<table>
<thead>
<tr>
<th>Field</th>
<th>Min occur</th>
<th>Max occur</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origination Date</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Date</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### 3.4.2. XMPP Message

Message ‘stanzas’ are used in XMPP to push information to other network entities such as users/contact and messaging servers. Messages are mostly used within the context of instant messaging between two parties, but can also be used within the context of public chat rooms, errors and event notification.

**Figure II-4 XMPP Message Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Min occur</th>
<th>Max occur</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Origination Date Field</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>Originator fields</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>from</td>
<td>1</td>
<td>1</td>
<td>&lt;message id='jcL_7' to='jane@localhost' type='chat' from='dana@localhost/Home'&gt;</td>
</tr>
<tr>
<td><strong>Destination address fields</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td>0</td>
<td>U</td>
<td>&lt;message id='jcL_7' to='jane@localhost' type='chat' from='dana@localhost/Home'&gt;</td>
</tr>
<tr>
<td><strong>Identification fields</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>0</td>
<td>1</td>
<td>&lt;message id='jcL_7' to='jane@localhost' type='chat' from='dana@localhost/Home'&gt;</td>
</tr>
<tr>
<td><strong>Informational fields</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.5. Markups: Blog: RSS 2.0

(Refer to: http://blogs.law.harvard.edu/tech/rss)

The concept of ‘Blogging’ is not new to the internet as it started with News Groups and Guestbook entries that allowed users to share their ‘thoughts’ with an online public audience. The creation of the RSS Standard (Really Simple Syndication) allowed ‘Blogging’ to play a much more significant role in the online messaging landscape. The RSS 2.0 syndication standard uses XML elements to enable a powerful unified public forum network which can be further enhanced through the use of externally defined namespace elements.

---

**Figure II-5 RSS 2.0 Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Min</th>
<th>Max</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td></td>
<td></td>
<td>&lt;rss version=&quot;2.0&quot;&gt;channel&lt;/rss&gt;</td>
</tr>
<tr>
<td>Origination Date Field</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LastBuildDate</td>
<td>0</td>
<td>1</td>
<td>&lt;lastBuildDate&gt;Tue, 16 Mar 2004 22:51:59 -0500&lt;/lastBuildDate&gt;</td>
</tr>
<tr>
<td>Originator fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Link</td>
<td>1</td>
<td>1</td>
<td>&lt;link&gt;<a href="http://www.devarticles.com">http://www.devarticles.com</a>&lt;/link&gt;</td>
</tr>
<tr>
<td>Informational fields</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>1</td>
<td>1</td>
<td>&lt;description&gt;mos_rss&lt;/description&gt;</td>
</tr>
<tr>
<td>Title</td>
<td>1</td>
<td>1</td>
<td>&lt;title&gt;Dev Articles - Programming Help and Tutorials for all development technologies&lt;/title&gt;</td>
</tr>
<tr>
<td>Language</td>
<td>1</td>
<td>1</td>
<td>&lt;language&gt;en-us&lt;/language&gt;</td>
</tr>
</tbody>
</table>
3.6. Markups: Calendar Event (VICAL)

(Refer to: http://www.imc.org/pdi/vcal-10.txt)

The emergence of diverse calendaring applications required the creation of a unifying standard that allows different devices to share scheduling data. The VCal standard is the text based standard that is used by most devices to communicate scheduling information. While this markup format is still quite primitive and relies on tab delimited text entries, it nevertheless provides a valuable insight onto the structure and requirements of calendar events.

Figure II-6 VICAL Event Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Min occur</th>
<th>Max occur</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origination Date</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1. Field Category Analysis: Common Denominators

While the markups explored in the previous section vary in presentation/terminology, we nevertheless were able to fit them fully within the field categories that we had derived from the RFC2822 Email Standard.
As a matter of fact, despite their various encoding styles (XML, plain text), variation in structure (single message document, multiple message document) and exchange methods (push, pull), the markup standards adhere to the field categories defined and many parallels can be drawn between their respective fields, highlighting several common denominators across markups. (e.g., all markups have at least one equivalent implementation of: ‘From’, ‘To’, ‘Subject’, ‘ID’ and body content).

Table II-3 tracks the various field mappings and compares them to the suggested IMF markup structure:

4.1.1. Origination Date:

All the standards we explore have at least one field that encodes the message creation date at the authors’ station. Dates are usually encoded using the universal time (Greenwich Mean Time) format.

4.1.2. Originator Field:

The most characteristic originator field we found was: ‘From’

Although we assume that most message markups would implement some equivalent of the ‘From’ field, our analysis shows that although the ‘From’ field is a common denominator across most the markups analyzed, there are significant differences in the manner through which it is implemented. As a matter of fact, while markups like the RFC2822 email standard, MSN IM and XMPP explicitly define ‘From’, markups such as Blog RSS 2.0 only imply ‘From’ equivalents such as the ‘Item Source’ and ‘Author’ fields.

Eventually, we were able to either derive or assume Originator Field equivalents for all markups, particularly in the case of the ‘From’ field.

4.1.3. Destination Field:

The most characteristic destination field we found was: ‘To’, nevertheless, several markups defined equally important fields such as ‘CC’, ‘BCC’.

As in the case of the originator fields, we have to derive or assume some of the ‘implied’ encoding, particularly in the case of Blog RSS 2.0 which is usually used as a ‘public’ bulletin board and therefore addresses a general anonymous audience. As we were intent on analyzing RSS 2.0 in the context of a message
based structure, we assumed that the ‘To’ field would point to the contact retrieving the blog post and would therefore be an ‘implied’ field.

4.1.4. Trace Field:
Trace fields are characterized mostly by time stamps encoding the message receipt date, i.e. the date the data was received and stored on the destination server. With the exception of the IM MSN transcript which uses the same timestamp for ‘Date’ and ‘DateTime’ to encode the date and time the message was saved to a local data store, all markups implement a ‘Receipt Date’ variant which differs from the ‘Date’(send date) field.

4.1.5. Identification Field:
All markups implement a unique message identifier and as such we were able to seamlessly map out identification fields.

4.1.6. Informational Fields:
Informational fields are mostly implemented as optional items across the various markups. As such, we selected subject as the most common denominator and were able to derive ‘Subject’ implementations from all markups.

4.1.7. Body Fields:
Body Fields define additional content enclosed by a message. Markups define message content as text, file attachments, and links to Internet resources or to local data stores. Some markups also implement additional logic within the body field, such as sending rules and text style definitions.
Overall, we identified 3 major common denominators: link, text and style.
<table>
<thead>
<tr>
<th>Table 3</th>
<th>Origination Date</th>
<th>Originator Fields</th>
<th>Destination Fields</th>
<th>Trace Fields</th>
<th>Identification Fields</th>
<th>Informational Fields</th>
<th>Body Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Date</td>
<td>From</td>
<td>To, CC, BCC</td>
<td>Received</td>
<td>message-id</td>
<td>subject, comments, keywords</td>
<td>MIME-Version, Content-type, charset, Content Message</td>
</tr>
<tr>
<td>MSN IM</td>
<td>Log: Date</td>
<td>Log: Logon Name, Message: Logon Name, Friendly Name</td>
<td>Message: Logon Name, Friendly Name</td>
<td>Log: DateTime, FirstSessionID, LastSessionID</td>
<td>Message: SessionID</td>
<td>Message: Time</td>
<td>Message: Text, File, Style</td>
</tr>
<tr>
<td>Blog RSS 2.0</td>
<td>Channel: LastBuildDate</td>
<td>Channel: Link Item: Source, Author</td>
<td>Channel: TTL, Item: PubDate</td>
<td>Channel: generator Item: guid</td>
<td>Channel: Description, Title, Language Item: Title, category, subject</td>
<td>Item: Link, Enclosure, Description</td>
<td></td>
</tr>
<tr>
<td>XMPP Presence</td>
<td>Date</td>
<td>From</td>
<td>To</td>
<td>stamp, thread</td>
<td>id</td>
<td>status, priority, type</td>
<td>—</td>
</tr>
<tr>
<td>XMPP Message</td>
<td>Date</td>
<td>From</td>
<td>To</td>
<td>stamp, thread</td>
<td>id</td>
<td>type, subject, composing</td>
<td>body</td>
</tr>
<tr>
<td>VCAL Calendar</td>
<td>Date</td>
<td>Organizer, Maltto</td>
<td>Attendee</td>
<td>DTSTAMP</td>
<td>UID</td>
<td>Location, Summary</td>
<td>Description, VALARM, RRULE, DTSTART, DTEND</td>
</tr>
<tr>
<td>IMF Message Package</td>
<td>MsgHeader: SendDate</td>
<td>MsgAddresses: FieldType: From</td>
<td>MsgAddresses: FieldType: To, CC, BCC</td>
<td>MsgHeader: ReceiptDate, ParentMsgID</td>
<td>MsgHeader: MsgID</td>
<td>MsgHeader: Subject, MsgType, Status</td>
<td>MessagePart, MessageRule</td>
</tr>
</tbody>
</table>
4.2. Integrated Message Schema

After identifying the major common denominators, we were able to decode 'blueprints' for a common framework which maps messages from across the various markups into one common format. The Integrated Message Schema (IMS) started to take shape:

4.2.1. Overview:

We viewed the Message as the centerpiece of all integration attempts, since it was a concept that is generic enough to incorporate any form of data exchange. As a matter of fact, we found it equally logical to label an email, blog post, calendar event, and IM chat thread: Message. A message, through its various field categories would be able to incorporate various metadata, and content types. Therefore, we could use a message to wrap a text exchange, document exchange (file, contact info), event exchange, or a combination of the previous three.

Extending our analysis of the markups, we compressed our categories further into four message wrappers that constitute an IMS Message Package: IMS Header, IMS Addresses, IMS Part, and IMS Rule (Figure II-7 IMS Message Package Constituents).

Furthermore, realizing that markups such as RSS 2.0 represented linked message lists, and intent on enabling IMS with message grouping capabilities, we defined a Project Wrapper: IMS Project (Figure II-8 IMS Project).

4.2.2. IMS Header

The IMS Header contains what can be referred to as the message metadata. In that respect, the wrapper encapsulates the following filed categories: ‘Origination Date’, ‘Trace’ fields, ‘Identification’ fields, ‘Information’ fields. The IMS Header main field would therefore be: MsgID, MessageType, Subject, ReceiptDate, and SendDate.

The IMS Header constitutes the main message component and is a required wrapper.

4.2.3. IMS Addresses
Given, the different media (Email, Blog, IM) encoded in the various markups and our desire to provide an all inclusive schema, we encapsulated the Originator Fields and Destination fields in the IMS Addresses wrapper. The wrapper defines an Address and an AddressType fields which used together can map all media types. Furthermore IMS Addresses defines FieldType which defines the address direction, i.e. ‘From’, ‘To’, ‘CC’, ‘BCC’...

The IMS Header in conjunction with IMS Addresses fully defines a Message Content.

All messages should define at least two IMS Addresses: ‘From’ address and ‘To’ address.

4.2.4. IMS Part

The IMS Part encapsulates the common Body Fields such as Enclosure, Link and Style. It therefore represents the message standard content such as text or document. The IMS Part references the IMS Header.

A message can contain zero or multiples occurrences of IMS Part.

4.2.5. IMS Rule

As in the case of the IMS Part, The IMS Rule derives its content from the Body Fields. Nevertheless, IMS Rule defines additional logic to be imbedded in a message such as event information (trigger date, start date, etc...), recurrence info, etc...

IMS Rule is intended to be an extendable wrapper that enables the IMS users to encode additional logic into their messages.

4.2.6. IMS Project

(Figure II-8 IMS Project)

The IMS Project, while not part of the IMS Message Package, is used to group a series of messages into a shareable entity. IMS Project uses constructs defined in RSS 2.0 Blog channels to wrap several Message Packages, through a 'linked list' package.

IMS Project will be extensively used in our I-LINK Client.
Figure II-7 IMS Message Package Constituents

ORIGINATION DATE
TRACE FIELDS
IDENTIFICATION FIELDS
INFORMATIONAL FIELD

ORIGINATOR FIELDS
DESTINATION FIELDS

BODY FIELDS

IMS MESSAGE HEADER
IMS MESSAGE ADDRESSES
IMS MESSAGE PART
MESSAGE RULE

IMS MESSAGE PACKAGE
Figure II-8 IMS Project
### Figure II-9 IMF Markup Mapping

<table>
<thead>
<tr>
<th>IMF Field Name</th>
<th>Email Fields</th>
<th>RSS 2.0 Fields</th>
<th>XMPP Message</th>
<th>XMPP Presence</th>
<th>MSN IM</th>
<th>VCAL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMF PROJECT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FriendlyName</td>
<td></td>
<td></td>
<td>Channel: Title</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AdminUserName</td>
<td></td>
<td></td>
<td>Channel: Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModifyDate</td>
<td></td>
<td></td>
<td>Channel: LastBuildDate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Online URL</td>
<td></td>
<td></td>
<td>Channel: Link</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IMF HEADER</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MsgBoxID</td>
<td>message-id</td>
<td>Item: guid</td>
<td>id</td>
<td>id</td>
<td>SessionID</td>
<td>UID</td>
</tr>
<tr>
<td>MessageType</td>
<td></td>
<td>Item: Category</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject</td>
<td></td>
<td>Item: Title</td>
<td>subject</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReceiptDate</td>
<td>Received</td>
<td>Item: PubDate</td>
<td>stamp</td>
<td>stamp</td>
<td>Log: DateTime</td>
<td>DTSTAMP</td>
</tr>
<tr>
<td>SendDate</td>
<td>Date</td>
<td>Channel: LastBuildDate</td>
<td>Date</td>
<td>Date</td>
<td>Log: Date</td>
<td>Date</td>
</tr>
<tr>
<td><strong>IMF MESSAGE ADDRESSES</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Type</td>
<td>From, To, CC, BCC</td>
<td>Item: Source, Author</td>
<td>From, To</td>
<td>From, To</td>
<td>LogonName</td>
<td>Organizer, Attendee</td>
</tr>
<tr>
<td>Address Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>From, To, CC, BCC</td>
<td>Item: Source, Author</td>
<td>From, To</td>
<td>From, To</td>
<td>LogonName</td>
<td>Organizer, Attendee</td>
</tr>
<tr>
<td>Status</td>
<td></td>
<td></td>
<td>composi</td>
<td>status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ModifyDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IMF MESSAGE PART</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part Type</td>
<td>Content-type</td>
<td>Item: Enclosure</td>
<td>type</td>
<td>type</td>
<td>Message, Invitation, InvitationResponse</td>
<td>Style</td>
</tr>
<tr>
<td>CharSet</td>
<td>CharSet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Content Message</td>
<td>Item: Description</td>
<td>body</td>
<td>Text</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>StorageAddress</td>
<td></td>
<td>Item: Link</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Part Addresses</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>IMF MESSAGE RULE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TriggerDate</td>
<td></td>
<td>Channel: TTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recurrence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>StartDate, End Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 11-10 IMS XML Sample

```xml
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfIMFMessagePackagexmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <IMFMessagePackage>
    <Message xmlns="http://tempuri.org/">
      <MsgID>1294</MsgID>
      <Subject>Lower blood pressure t wuwckzf ocq</Subject>
      <Received>2004-04-09T05:33:23.0000000-04:00</Received>
      <Sent>2004-04-09T01:30:12.0000000-04:00</Sent>
      <MessageType>Message</MessageType>
      <UserName>td4</UserName>
      <ParentID>0</ParentID>
      <inbound>true</inbound>
      <MsgSize>1771</MsgSize>
    </Message>
    <Addresses xmlns="http://tempuri.org/">
      <IMFMessageAddress>
        <Address>tdajani@cleartag.com</Address>
        <MediaType>Email</MediaType>
        <MsgID>0</MsgID>
        <ModifyDate>0001-01-01T00:00:00.0000000-05:00</ModifyDate>
        <DeleteDate>0001-01-01T00:00:00.0000000-05:00</DeleteDate>
        <FieldType>To</FieldType>
        <IsUser>false</IsUser>
      </IMFMessageAddress>
      <IMFMessageAddress>
        <Address>x22rfmgy@mailcity.com</Address>
        <MediaType>Email</MediaType>
        <MsgID>0</MsgID>
        <ModifyDate>0001-01-01T00:00:00.0000000-05:00</ModifyDate>
        <DeleteDate>0001-01-01T00:00:00.0000000-05:00</DeleteDate>
        <FieldType>From</FieldType>
        <IsUser>false</IsUser>
      </IMFMessageAddress>
    </Addresses>
    <Parts xmlns="http://tempuri.org/"/>
  </IMFMessagePackage>
</ArrayOfIMFMessagePackage>
```
Chapter III  Integrated Message Framework (IMF)

1. What is IMF?
In the previous chapter we introduced the Integrated Message Schema (IMS) as a message wrapper that enables a unified markup of different media messages. The Integrated Message Framework (IMF) uses IMS as a foundation for a centralized messaging framework that allows the publication, retrieval and manipulation of various media messages. Furthermore, IMF enables a ‘project oriented’ access to message repositories.
In the sections to follow, we will explore in depth all the IMF components.

2. IMF Objectives

2.1. Unified Text and Document Messaging Format
IMF provides a unified format for encoding all types of text and document messages. Whether a message is an Email, a blog post, or a calendar event, IMF parses it and packages it as a serializable integrated message package therefore eliminating the barrier between message medium, format and presentation.

2.2. Intelligent Messaging
IMF enables messaging features that leverage capabilities inherent in individual existing media. In that respect IMF allows message delivery and dispatch to be scheduled and prioritized regardless of the destination or source medium. It also introduces contact presence awareness across media formats, therefore enabling the implementation of Intelligent Address Book and Message Dispatching.

2.3. Independence of Media Type
Through the incorporation of several delivery channels (web services, blog RSS, and TCP calls) IMF detaches itself from message presentation and leaves a wide open range of client implementations/integration. As a matter of fact, IMF messages can be accessed from blog aggregators, web clients and windows applications alike.
2.4. Centralized Messaging

IMF capitalizes on the fact that online storage space is becoming less and less expensive in order to provide a centralized consolidated message repository. As such messaging information can be accessed from a wide variety of clients.

3. IMF Environment

IMF is implemented as a centralized core accessed by various ‘consumer’ clients. These clients access the framework through various network channels which require three main server components: Internet Server, Database Server and Storage Server

3.1. Web Server

Whether access is through a LAN or the Internet, all the IMF components are accessed through standard TCP/IP protocols and are channeled through the Microsoft Internet Information Server (IIS). IMFServices is defined as an IIS application and provides the access point to all the IMF components, be it direct web access, web services subscription, or RSS feed requests.

Furthermore, IIS, through the FrontPage Server Extensions enables remote access through Microsoft Visual Studio to the core IMF libraries therefore significantly improving framework debugging, new feature deployment and component upgrades.

3.2. Database Server

IMF relies extensively on a centralized message (data) repository in order to deliver its integration promises. As such, while access to IMF is handled by IIS, data is stored in a central Microsoft SQL Server Database.

SQL Server provides a powerful platform for implementing the IMF data model particularly through its support for stored procedure, triggers, and transactions.

Furthermore, SQL server provides a strong backup engine.

3.3. Storage Server

In addition to message ‘metadata’ stored in the SQL Server Database, IMF requires a physical storage server that stores shared messages, and temporary
message copies that can either become permanent or be downloaded by consumer clients such as I-LINK (Chapter V.1).

4. IMF Building Blocks
IMF is implemented as a modular architecture based on several basic logical entities that are extrapolated from the IMS. These entities are: user, project, contact, address, account and message:

4.1. IMF User Definition
An IMF user is the basic security unit for accessing the IMF functionality. As such, a user account enables its owner to use the IMF Web Services to store, retrieve and manage his centralized message repository. Furthermore it allows him/her to create and manage multiple sharing projects.

While the IMF User provides an entry point to the full IMF functionality, it is not the sole repository access point. As a matter of fact, through the combined use of projects and IMF’s generic message access functionality, non users are provided access to project RSS feeds or public web project access.

4.2. IMF Project Definition
An IMF project is a logical dynamic entity that allows the grouping of several contacts, messages and accounts into a shareable group. The project is used to share resources between various contacts through multiple media such as email and blog or project management clients such as I-LINK.

IMF project information is structured in conformance with the IMS project. IMF Project information can be exchanged as a TCP transmitted XML stanza, RSS 2.0 feed, .Net Serialized Type, web display or email message.

4.3. IMF Contact Definition
An IMF contact is a logical entity that groups several media addresses under the umbrella of an owning contact person. The current IMF implementation defines a basic information structure made up of a contact Friendly Name, Company, and Mailing Address, along with multiple possible messaging addresses. An IMF Contact can be used in conjunction with an IMF User to provide additional user
information. It can also be defined as a user owned contact (a user address book entry), or a project owned contact (a project address book entry). Furthermore, the IMF contact features a prioritized address list which allows the sorting of access information according to the owner’s preferences. Such a feature enables a powerful IMF feature: Dispatch Preferences (Chapter III.6.4)
IMF Contact information can be exchanged as a TCP transmitted XML stanza, RSS 2.0 feed, .Net Serialized Type, Web display or Email message.

4.4. IMF Message Definition
IMF Messages are the core constituents of IMF.
A message is a composite data package that has a type (document/file, message, conversation, event, system), an origin, one or multiple destinations/targets, a payload (attachments, body text, conversation threads), a subject, a postmark (date sent, date received by server) and a specific processing logic.
Within the context of IMF, messages refer to IMF Message Package as it is the main data exchange currency.
Within that context a message represents an email message, a calendar event, a blog entry, an instant message transcript, a file share, a file transfer, an event, error or task notification, or a system command.
An IMF Message maps the IMS Package structure and is therefore built from the following components: Header, Message Addresses, Message Parts, and Message Rules.
IMF message information is structured in conformance with the IMS Message Package.
IMF Message information is exchanged as a TCP transmitted XML stanza, RSS 2.0 feed, .Net Serialized Type, Web display or Email message.
Furthermore, IMF Messages are shared and exchanged through IMF Projects.

4.5. IMF Address Definition
An IMF address defines a basic address entry. The current IMF implementation defines an address as a combination of an address string (email, URL, etc...), and a media type (email, blog, IM, and system). Addresses can be paired up with other IMF entities to create complex entities such as Message Address and Contact
Addresses. IMF Address entities are used internally by the system and are almost always package into complex entities before any network exchange. System addresses are created by the IMF framework seamlessly upon the creation of IMF user accounts.

4.6. IMF Account Definition
An IMF account extends an IMF Address through providing retrieval and publication access information. In other words, an Account entity defines such information as Inbound URL (ex: POP3 server address), Inbound port, Outbound URL and outbound port, as well the authentication credentials required for accessing these servers.
The IMF Account media type (Email, Blog, IM, and System) is determined by the address which the account extends.
While IMF accounts can be defined for projects, they nevertheless, cannot be accessed through RSS feeds for security reasons.

5. Integrating Message Media
The key to IMF’s success is its ability to integrate the various messaging media in the market into a comprehensive framework. As such, IMF enables messaging capability that is compatible with most mainstream messaging platforms:

5.1. Email
IMF provides full email functionality through its integration of the POP3 and SMTP protocols within its core management classes. As such, IMF allows clients to send and retrieve IMF Message Package through the IMF POP3 and SMTP Mail connectors. IMF also caters for email file attachments.

5.2. Blog
IMF fully integrates the RSS 2.0 blog standard and as such provides the capability to retrieve blog posts and publish IMF Messages and Projects as RSS 2.0 blog feeds.

5.3. Instant Messaging
Many Instant Messaging standards are available on the market. As such, integrating all IM formats requires extensive ‘translation work’. IMF has prepared
the groundwork for such integration through implementing its own message notification and instant transfer mechanism.

Such a mechanism currently enables IMF clients to make use of its System IM while enabling future IM platforms integration.


The previous sections have defined basic structures that were ‘derived’ from our analysis of the various media markups.

While these structures provide a solid foundation for the core IMF functionality, they do not take into considerations the particularities/advantage inherent in individual markup formats.

We therefore found it appropriate to explore several concepts encountered throughout our analysis and highlight the advantages and challenges of incorporating them as ‘Intelligent’ Framework features:

6.1. Presence

Presence can be defined as the network availability of a user or resource. It is particularly encountered in instant messaging applications and standards (XMPP, MSN).

As we ventured into our analysis of the MSN messaging transcripts and the XMPP protocol we realized the value of incorporating the concept of presence into the IMF. As a matter of fact, we found that the concept of Presence lends itself well for superimposition over the IMF building blocks in order to provide a powerful presence enabled integrated framework.

6.1.1. Strategy

The main challenge for presence enabled architectures is to keep track of user and resource network status through time and across various media. Example: a user can have an email address and a MSN Messenger account which maintains the user’s status. While the MSN messenger is ‘aware’ of the user’s status, it does not provide a structured mechanism for sharing such info with other address (MIT email address for instance).
As such, our first presence implementation strategy was to provide an intrinsic IMF link between addresses affiliated with one common resource or user. This approach allowed presence information gathered about a single address to be shared with all affiliated addresses, i.e. if IMF was aware of a user’s IMF status, any address affiliated to the MSN address such as an email would be aware of the user’s presence (Figure III-1 IMF Presence Relay).

The second approach involves leveraging presence information in order to provide an efficient message dispatching procedure. As a matter of fact, using the structured presence data, IMF is able to determine the fastest message notification channel for a specific user (Figure III-2 IMF Presence Informed Dispatch).
6.1.2. IMF Roster and IMF Presence

An IMF Roster Block encapsulates all the presence information available for a certain user. The IMF Roster is compiled in the IMF by gathering information from the various IMF Building blocks.

The IMF Roster wraps an IMF Contact object along with an IMF Presence block which contains network status information (online, offline), current presence (away, be right back, etc.).

6.1.3. Future Implementations

Presence information can be applied to most network resources such as shared folders, printers, fax machines, etc... As such we anticipate being able
to leverage the value of presence through IMF in order to achieve an intelligent and highly aware media network.

6.2. File Transfer and Sharing

Our initial reaction was to analyze file transfer and file sharing from a markup point of view. In other word, we considered exploring different file metadata structures within the context of media messaging. Nevertheless, we soon realized that the above topics are implicitly encapsulated in most messaging markups.

As a matter of fact, from email attachments, MSN file transfer to Outlook event file attachment, most messaging markups cater for some form of file messaging. That conclusion leads us to an alternative approach which consisted of excluding file metadata from our analysis and instead encapsulates files within IMF Message Parts.

In that respect file transfer and sharing can be superimposed over the IMF building blocks to extend the IMF functionality. The combined framework allows for a versatile file enabled message framework:

6.2.1. IMF Document Message and IMF Part

An IMF Document Message represents a Message Package of type document. As such, the main purpose of the IMF Document Message is to encapsulate a physical file and make it available to one or many recipients. Information about file contents, such as storage location and file types are stored in the message’s IMF Part.

6.2.2. File Sharing

A shared file is structured as an IMF Document Message addressed to several contacts. The physical file is stored on the central server and is referenced through an IMF Part. Recipients (representing the individuals with whom the file is shared) are notified via a Message Package and can download the file from the central server. (Figure III-3 IMF File Sharing)
6.2.3. File Attachment

Attachments that we commonly encounter with email are encapsulated within the Message Package through IMF Parts. An incoming email for instance is parsed into a Message Package of type message, with all attachments being referenced by IMF Parts.

File attachments are stored on the central server and can be requested/downloaded by the message recipients. (Figure III-4 File Attachment)
6.2.4. File Transfer

File Transfer occurs within the context of an Instant Messaging Session which is encapsulated in an IMF Message Package of type conversation. A transferred file is stored on the central server and the file recipients within the IM Session are immediately notified of incoming invitation. (Figure III-5 IMF File Transfer)
6.2.5. Future Implementations

The logical extension to the IMF File capabilities would be to enable peer to peer file access mediated by the IMF Framework. Such capabilities require clients that are capable of communicating directly, and of crossing firewall restrictions.

6.3. Message Handling Rules

As we explored the various markup structures, we came to appreciate the value of leveraging concepts such as scheduled events, spam blocking, and address management. As such, we pinpointed several features that would bring in unique
value to the IMF: delayed dispatch, block sender and forward message. While these features were not implanted within the I-LINK client’s second spiral they are nevertheless catered for in full within IMF.

6.3.1. IMF Message Rule

The IMF Rule is a central component of the IMF Handling functionality. An IMF Rule is encapsulated within an IMF Message Package and contains specific processing logic.

6.3.2. IMF Address Rule

The IMF Address Rule is a central component of the IMF Handling functionality. An IMF Address Rule is affiliated to an IMF Address and provides specific processing logic for messages received to and from that address.

6.3.3. Delayed Dispatch

While most messages are usually dispatched by the server immediately upon being sent by a user, there are instances where a message might require a delayed dispatch. Ex: in a team development environment a supervisor might require project updates at a specific time.

Delayed dispatches are implemented through the IMF Rule trigger date field and can be applied to any IMF Message Package.

6.3.4. Block Sender

Through the IMF Address Rules, IMF enables us to block an IMF Address from sending/ receiving messages. Such capability leverage IMF’s Address lists in order to propagate the block rule to all addresses affiliated with the blocked IMF Address.

6.3.5. Forward Message

As in many mail clients, the IMF enables users to set Address Rules that specify a forwarding address for messages incoming to the rule’s associated IMF Address.

6.4. Dispatch Preferences

While Message handling rules allow for intelligent message content processing, we found that a successful integrated environment would require seamless handling of messages. As such, a user sending a message to certain recipient should not be
concerned (unless he explicitly decides to) with the media through which the message is dispatch.

In order to achieve such functionality we supplemented base blocks such IMF Accounts, and IMF Addresses with priority fields. These fields are specified by the block’s owner to reflect his/her dispatch preferences. The IMF would then process outgoing messages through the preset priority list.

**Ex:** a user can send a message to a recipient. The recipient is configured with an email address (priority 1) and an IM address (priority 2). The IMF processor would determine the message dispatch destination through examining the address priority list.
Chapter IV IMF Architecture

1. Overview
IMF is organized as a 5 tier service oriented architecture. At the base of the framework reside the database and file storage (2). The data model is then abstracted, encapsulated and operated on by a set of base classes, management classes, media connectors and media parsers: IMF Classes (3). The IMF classes provide internal processing functionality which is delivered through either of two interaction tiers: IMF Web Services (4), and IMF Generic Message Services (5) and consumed by various remote clients (6).

2. IMF Database Procedures
IMF is implemented around a MS SQL Database. While the IMF core classes interact directly with the database, we have chosen to implement resource intensive queries
through stored procedures while implementing several data table triggers that perform basic framework business logic and cleanup operations.

3. IMF Core Classes

3.1. IMF Types

IMF Types are public serializable types that are used for all data exchanges across IMF. IMF Types are containers for data which are operated upon and delivered through a set of management classes: IMF Type Managers (d) and external resource connectors: IMF Media Connectors (e).

IMF Types follow closely the IMF Data Model structure and implement the .Net ISerializable Interface which requires them to be public types.

3.1.1. IMF Session

IMF Session encapsulates user session information such as the calling IP and Port. IMF Session is used in the authentication process and in timeout and notification operations.

![Figure IV-2 IMF Session](image)

<table>
<thead>
<tr>
<th>IMFSession</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Login string</td>
</tr>
<tr>
<td>+PassKey string</td>
</tr>
<tr>
<td>+IP string</td>
</tr>
<tr>
<td>+Port string</td>
</tr>
<tr>
<td>+IMFSession();</td>
</tr>
<tr>
<td>+IMFSession(in login string in IP string in Port string)</td>
</tr>
<tr>
<td>+IMFSession(in login string in IP string in Port string in passkey string)</td>
</tr>
</tbody>
</table>

3.1.2. IMF User

IMF User encapsulates information related to a system user and is used mainly in authentication and presence verification operations. IMF User structure is based on the IMF User Definition (Chapter III .4.1)

![Figure IV-3 IMF User](image)

<table>
<thead>
<tr>
<th>IMFUser</th>
</tr>
</thead>
<tbody>
<tr>
<td>- _Password string</td>
</tr>
<tr>
<td>+CreateDate DateTime</td>
</tr>
<tr>
<td>+UserName string</td>
</tr>
<tr>
<td>+Password() string</td>
</tr>
<tr>
<td>+IMFUser()</td>
</tr>
<tr>
<td>+IMFUser(in login string in password string)</td>
</tr>
<tr>
<td>+CompareTo(in ob object) : int</td>
</tr>
</tbody>
</table>
3.1.3. IMF Project

IMF Project encapsulates IMF Project information. IMF Project structure is based on the IMF Project Definition (Chapter III.4.2).

<table>
<thead>
<tr>
<th>IMFProject</th>
</tr>
</thead>
<tbody>
<tr>
<td>+AdminUserName : string</td>
</tr>
<tr>
<td>+OnlineURL : string</td>
</tr>
<tr>
<td>+RSSURL : string</td>
</tr>
<tr>
<td>+FriendlyName : string</td>
</tr>
<tr>
<td>+IsPrivate : bool</td>
</tr>
<tr>
<td>+ProjectID : int</td>
</tr>
<tr>
<td>+ModifyDate : DateTime</td>
</tr>
<tr>
<td>+IMFProject()</td>
</tr>
<tr>
<td>+IMFProject()</td>
</tr>
<tr>
<td>+IMFProject()</td>
</tr>
<tr>
<td>+IMFProject()</td>
</tr>
<tr>
<td>+IMFProject()</td>
</tr>
<tr>
<td>+CompareTo()</td>
</tr>
</tbody>
</table>

3.1.4. IMF Contact

IMF Contact encapsulates IMF Contact information. IMF Contact structure is based on the IMF Contact Definition (Chapter III.4.3).

<table>
<thead>
<tr>
<th>IMFContact</th>
</tr>
</thead>
<tbody>
<tr>
<td>+IsUser : bool</td>
</tr>
<tr>
<td>+Show : bool</td>
</tr>
<tr>
<td>+UserName : string</td>
</tr>
<tr>
<td>+ContactName : string</td>
</tr>
<tr>
<td>+Company : string</td>
</tr>
<tr>
<td>+MailAddress : string</td>
</tr>
<tr>
<td>-_ContactID : int</td>
</tr>
<tr>
<td>+Main : bool</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+IMFContact()</td>
</tr>
<tr>
<td>+ContactID()</td>
</tr>
</tbody>
</table>

3.1.5. IMF Address

IMF Address is used in the exchange of address information, message dispatch and message fetching. IMF Address is also used as a building block for other IMF complex types used in message and contact information exchange such as IMF Message Address and IMF Contact Address. IMF Address structure is based on the IMF Address Definition (Chapter III.4.5).
3.1.6. IMF Account

IMF Account is used in the exchange of user account information, message dispatch and message fetching. IMF Account structure is based on the IMF Account Definition (Chapter III.4.6).

3.1.7. IMF Message Header

IMF Message Header contains the basic message header information and is used as the entry point for message access. IMF Message Header is used in complex types such as message package and notification package. (Figure IV-8 IMF Message Header)
3.1.8. IMF Message Address

IMF Message Address packages an IMF address as part of a message. It is used as an association class between IMF Address and IMF Message in order to encode message senders and recipients.

3.1.9. IMF Message Contact

IMF Message Contact packages an IMF contact as part of a message. It is used as an association class between IMF Contact and IMF Message in order to encode message senders and recipients for intelligent IMF dispatching.
3.1.10. IMF Message Package

IMF Message Package represents the main messaging component and encapsulates several base types such as IMF Message Header, IMF Message Addresses, IMF Message Parts, IMF Message Contact and IMF Message Rules (Figure II-7 IMS Message Package).

<table>
<thead>
<tr>
<th>IMFMessagePackage</th>
</tr>
</thead>
<tbody>
<tr>
<td>+Message : IMFMessage</td>
</tr>
<tr>
<td>+Contacts : IMFMessageContact[] = null</td>
</tr>
<tr>
<td>+Addresses : IMFMessageAddress[]</td>
</tr>
<tr>
<td>+Parts : IMFPart[] = null</td>
</tr>
<tr>
<td>+IMFMessagePackage()</td>
</tr>
<tr>
<td>+IMFMessagePackage()</td>
</tr>
<tr>
<td>+IMFMessagePackage()</td>
</tr>
<tr>
<td>+IMFMessagePackage()</td>
</tr>
<tr>
<td>+IMFMessagePackage()</td>
</tr>
</tbody>
</table>

3.1.11. IMF Message Part

IMF Message Part encapsulates message body contents.

<table>
<thead>
<tr>
<th>IMFMessagePart</th>
</tr>
</thead>
<tbody>
<tr>
<td>+PartID : int</td>
</tr>
<tr>
<td>+Text : string</td>
</tr>
<tr>
<td>+Charset : string</td>
</tr>
<tr>
<td>+Storage : string</td>
</tr>
<tr>
<td>+Type : string</td>
</tr>
<tr>
<td>+ModifyDate : DateTime</td>
</tr>
<tr>
<td>+IMFPart()</td>
</tr>
<tr>
<td>+IMFPart()</td>
</tr>
<tr>
<td>+IMFPart()</td>
</tr>
<tr>
<td>+IMFPart()</td>
</tr>
<tr>
<td>+IMFPart()</td>
</tr>
</tbody>
</table>

3.1.12. IMF Message Rule

IMF Message Rule encapsulates message processing rules that are parsed by IMF (Chapter III. 6.3.1).

3.1.13. IMF Notification Package

IMF enables message passing through direct TCP connections and as such, the IMF Notification Package is used in order to transfer serialized IMF Types across direct TCP Connections.
3.1.14. IMF Presence

IMF Presence encapsulates user network status and presence messages.

3.1.15. IMF Roster

IMF Roster is an association class that encapsulates a user’s contact information along with his/her presence information.

3.2. Type Managers

Type Managers are private IMF classes that provide the core IMF functionality:

3.2.1. IMF Account Manager

IMF account manager provides core account management operations such as adding, updating and deleting accounts. The account manager also retrieves and updates account priority (Figure IV-16 IMF Account Manager).
3.2.2. IMF Contact Manager
IMF contact manager handles core contact management operations such as adding, updating and deleting contacts. It also performs association operations that link addresses to a contact. Furthermore, the Contact Manager enables the update and retrieval of contact address lists, and address priority lists. (Figure IV-17 IMF Contact Manager).

3.2.3. IMF Address Manager
IMF address manager handles core address management operations such as adding, updating and deleting addresses. It also performs base checks in order to avoid duplicate address data and database inconsistencies (Figure IV-18 IMF Address Manager).

3.2.4. IMF Message Manager
IMF message manager handles core message management operations such as adding, updating and deleting message packages, message addresses, message contacts, message parts and message rules. It also performs core message retrieval operations that are essential for search operations (Figure IV-19 IMF Message Manager).

3.2.5. IMF Presence Manager
IMF Presence Manager handles presence retrieval. The Presence manager encapsulates information into presence rosters, update user presence information and makes presence change notification calls (Figure IV-20 IMF Presence Manager).

3.2.6. IMF User Manager
IMF User Manager handles user creation, authentication, password change. It also retrieves a user’s main contact information and project information (Figure IV-21 IMF User Manager).

3.2.7. IMF Project Manager
IMF project manager handles core project management operations such as adding, updating and deleting projects. If allows adding and deleting messages, contacts and users to and from a project (Figure IV-22 IMF Project Manager).

3.2.8. IMF Session Manager
IMF Session Manager handles session management. It is used by internal server processes in order to check for expired session, update session access and create new client sessions (Figure IV-23 IMF Session Manager).

Figure IV-16 IMF Account Manager

-connection * command

SqlConnection SqlCommand

Type Managers: IMF_AccountManager
-sqlConnection: SqlConnection
-sqlCommand: SqlCommand

IMF_AccountManager

-CheckExist(in account : IMFAccount) : bool
-GetMaxPriority(in account : IMFAccount) : int
-GetMaxPriority(in project : IMFProject) : int
-GetNewPriority(in account : IMFAccount) : int
-GetNewPriority(in project : IMFProject) : int
+Add(in account : IMFAccount) : bool
+Add(in account : IMFAccount in project : IMFProject) : bool
+DBInsert(in account : IMFAccount in project : IMFProject) : bool
+Delete(in account : IMFAccount) : bool
+DeletePriority(in account : IMFAccount in priority : int) : bool
+Delete(in account : IMFAccount in project : IMFProject) : bool
+GetAccounts(in project : IMFProject) : IMFAccount
+GetAccounts(in user : IMFUser in type : string) : IMFAccount
+GetAccounts() : IMFAccount

Web Services: InitHandler
-components : IContainer = null
+InitHandler()
-InitializeComponent()
#Dispose()
+Login()
+Logout()
+AddUser()
+GetProjects()
+GetProjectContacts()
+GetProjectRosters()
+UpdateProject()
+CreateProject()
+UnsubscribeProject()
+AddProjectRequest()
+AddProjectMessage()
+RemoveProjectContact()
+GetUserRoster()
+UpdatePresence()
+ChangePassword()
+GetContactRoster()
+GetContacts()
+GetContactAddresses()
+UpdateContact()
+AddContact()
+DeleteContactAddress()
+DeleteContact()
+AddContactAddress()
+UpdatePriority()
+GetUserAccounts()
+GetProjectAccounts()
+AddAccount()
+AddProjectAccount()
+RemoveProjectAccount()
+DeleteAccount()
+Test()
+CheckRunning()
+TestPack()
+TestMail()
+Mail()
+SCheck()
+SCheck2()
+sendtest()
+GetUserMessages()
+GetProjectMessages()
+GetProjectDocuments()
+AddMessagePackage()
+AddProjectPackage()
Figure IV-17 IMF Contact Manager

- **SqlConnection**
- **SqlCommand**

Type Managers:
- IMF ContactManager
  - Connection

- IMF ContactManager
  - GetInsert(): int
  - Add(in contact: IMFContact): int
  - Update(in contact: IMFContact): boolean
  - Delete(in contact: IMFContact): boolean
  - GetContact(in contactId: int): IMFContact
  - GetMaxPriority(in contactId: int): int
  - GetNewPriority(in contactId: int): int
  - UpdatePriority(in address: IMFContactAddress): boolean
  - UpdatePriority(in address: IMFContactAddress, in priority: int): boolean
  - DeleteAddress(in contactId: int, in address: string): boolean
  - GetAddress(in contact: IMFContact, in priority: int): IMFContactAddress
  - GetFirstAddress(in contact: IMFContact): IMFContactAddress
  - GetPriorityList(in contact: IMFContact): IMFContactAddressList

- IMF Contact
  - IsUser: boolean
  - Show: boolean
  - UserName: string
  - Company: string
  - MailAddress: string
  - ContactID: int
  + IMFContact()

- IMF ContactAddress
  - Address: string
  + IMFContactAddress()

- IMF Address
  - Address: string
  + IMFAddress()
**Figure IV-18 IMF Address Manager**

- **SqlConnection**
- **SqlCommand**

```
IMFAddressManager::IMFAddressManager()
+ CheckExists(in address : string) : bool
+ CheckExists(in address : IMFAddress) : bool
+ Add(in address : string, in media : string) : bool
+ Delete(in address : string)
+ Delete(in address : IMFAddress)
```

```
IMFAddress
+ Address : string
+ MediaType : string
+ FriendlyName : string
+ IMFAddress()
+ IMFAddress()
+ IMFAddress()
```

**InitHandler**
```
+ InitHandler()
- InitializeComponent()
- Dispose()
+ Logon()
+ Login()
+ AddUser()
+ GetProjects()
+ GetProjectContacts()
+ GetProjectRoster()
+ UpdateProject()
+ CreateProject()
+ UnsubscribeProject()
+ AddProjectContact()
+ AddProjectMessage()
+ RemoveProjectContact()
+ GetUserRoster()
+ UpdateRoster()
+ ChangePassword()
+ GetContactRoster()
+ GetContacts()
+ GetContactAddresses()
+ UpdateContact()
+ AddContact()
+ DeleteContactAddress()
+ DeleteContact()
+ AddContactAddress()
+ UpdatePriority()
+ GetUserAccounts()
+ GetProjectAccounts()
+ AddAccount()
+ AddProjectAccount()
+ RemoveProjectAccount()
+ DeleteAccount()
+ Test()
+ CheckRunning()
+ TestPack()
+ TestMail()
+ Mail()
+ SCheck()
+ SCheck2()
+ sendtest()
+ GetUserMessages()
+ GetProjectMessages()
+ GetProjectDocuments()
+ AddMessagePackage()
+ AddProjectPackage()
```
Figure IV-19 IMF Message Manager

-sqlConnection1 : SqlConnection
-sqlCommand1 : SqlCommand

-IMF_MessageManager()  
+GetInsert : int  
+Add(message : IMFMessage) : int  
+DBInsert(message : IMFMessage) : int  
+AddPart(message : IMFMessagePart) : int  
+AddParts(message : IMFMessagePart[]) : int  
+AddContacts(message : IMFMessageContact) : bool  
+AddAddress(message : IMFMessageAddress) : bool  
+AddAddress(message : IMFMessageAddress[]) : bool  
+AddPackage(message : IMFMessagePackage) : int  
+Update(message : IMFMessage) : 
+Delete(message : IMFMessage)  
+GetMessages(user : IMFUser, inbound : bool, top : int, in flag : bool) : IMFMessagePackage[]  
+GetMessages(user : IMFUser, in address : IMFAddress, type : string, inbound : bool) : IMFMessagePackage[]  
+GetMessagesAfter(user : IMFUser, since : DateTime, inbound : bool) : IMFMessagePackage[]  
+GetMessagesBefore(project : IMFProject, before : DateTime, inbound : bool) : IMFMessagePackage[]  
+GetChildren(message : IMFMessage) : IMFMessagePackage[]  

IMFProject  
+AdminUserName : string  
+OnlineURL : string  
+RSSURL : string  
+FriendlyName : string  
+IsPrivate : bool  
+ProjectID : int  
+ModifyDate : DateTime  
+IMFProject()  
+IMFProject()  
+IMFProject()  
+IMFProject()  
+IMFProject[]()  

IMFUser  
+Password : string  
+CreateDate : DateTime  
+UserName : string  
+Password()  
+IMFUser()  
+IMFUser()  
+IMFUser()  
+IMFUser()  
+CompareTo()  

IMFMessageContact  
+Contact : IMFContact  
+Field : string  
+IMFMessageContact()  
+IMFMessageContact()  
+IMFMessageContact()  

IMFMessagePart  
+PartID : int  
+Subject : string  
+Charset : string  
+IMFMessagePart()  
+IMFMessagePart()  
+IMFMessagePart()  
+IMFMessagePart()  

IMFMessageAddress  
+Address : IMFAddress  
+ModifyDate : DateTime  
+Address()  
+Address()  
+Address()  
+Address()  

IMFMessageHeader  
+MsgID : int  
+Text : string  
+Origin : string  
+From : string  
+IMFMessage()  
+IMFMessage()  
+IMFMessage()  
+IMFMessage()  

IMFMessage  
+ParentID : int  
+Inbound : bool  
+MSGSize : int  
+Origin : string  
+Status : string  
+IsUser : bool  
+IMFMessageAddress()  
+IMFMessageAddress()  
+IMFMessageAddress()  
+IMFMessageAddress()  

IMFMessagePackage  
+Message : IMFMessage  
+Contacts : IMFMessageContact[] = null  
+Addresses : IMFMessageAddress[] = null  
+Parts : IMFMessagePart[] = null  
+IMFMessagePackage()  
+IMFMessagePackage()  
+IMFMessagePackage()  
+IMFMessagePackage()  

62
Figure IV-21 IMF User Manager

Type Managers: IMF_UsersManager

-IMF_UsersManager()
-Authenticate(in login: string, in pwd: string): bool
-Authenticate(in user: IMFUser): bool
-Authenticate(in user: IMFUser, in updatepresence: bool): bool
-CheckExists(in login: string): bool
-Add(in user: IMFUser): bool
-Add(in user: IMFUser, in contact: IMFContact, in address: string): bool
-DBInsert(in login: string, in pwd: string): bool
-DBInsert(in user: IMFUser): bool
-Delete(in user: IMFUser)
+GetMain(in user: IMFUser): IMFContact
+GetMain(in login: string): IMFContact
+GetContacts(in user: IMFUser): IMFContact[]
+GetProjects(in user: IMFUser): IMFProject[]
+GetProjects(in user: string): IMFProject[]
+GetUsers(): IMFUser[]
+GetCurrentUsers(): IMFUser[]

IMFProject
+AdminUserName: string
+OnlineURL: string
+RSSURL: string
+FriendlyName: string
+lsPrivate: bool
+ProjectID: int
+ModifyDate: DateTime
+IMFProject() + IMFProject() + IMFProject()
+CompareTo() + CompareTo() + CompareTo()
### 3.3. Media Connectors

Media Connectors handle end point connections to all types of media such as POP3 Email, Blogs and IM platforms.

#### 3.3.1. Mail

The Mail connector handles connections to POP3 and SMTP email server. It allows sending and receiving emails through IMF Accounts of Type email. The Mail Connector is called upon periodically by the IMF Server process in order to...
fetch incoming mail. The connector is also used for retrieving email messages count, sending emails and packaging email messages as IMF Message Packages through combined calls to the IMF Mail Parser and IMF Message Manager.

3.3.2. Blog
The Blog connector handles connections to blog URLs. It allows retrieving blog posts through IMF Accounts of Type Blog. The blog connector is called upon by the IMF Server process periodically in order to fetch blog posts. The connector is also used for retrieving blog post counts, and packaging blog messages as IMF Message Packages through combined calls to the IMF Blog Parser and IMF Message Manager.

3.3.3. Notification
The Notification connector is used in conjunction with the I-LINK Notification Listener in order to send TCP notifications. The notifications use the IMF Notification Package in order to push to the I-LINK client Message, and Project Notifications. It is also an essential component of the IMF System IM communication.

3.3.4. IM
While the IM connector is not fully implemented in spiral 2 of the IMF/I-LINK client, it nevertheless has full framework support for future implementation. The IM connector provides end point access to various messaging platforms such as MSN IM, Yahoo IM and Jabber Messengers.

3.4. Media Parsers
3.4.1. Mail Parser
The Mail parser is called upon by the Mail Connector in order to parse the original content of incoming email messages and encodes them in an IMF Message Package format. It allows parsing email headers, and processing message attachments.

3.4.2. Blog Parser
The Blog parser is called upon by the Blog Connector in order to parse the RSS 2.0 Feeds and encode its content into IMF Message Package format.

3.4.3. IM Parser
In order to provide support for various existing IM standards, we will need to provide specific IM parsers for individual IM standards in future implementations.

3.5. Utilities
3.5.1. IMF Server Process
The IMF server is a session related process which handles IMF ‘housekeeping’ and management functions, such as calling Mail connectors, Blog connectors, Notification connectors and Session managers.

3.5.2. IMF Utilities
The IMF Utilities provides global string manipulation, data encryption and database authentication methods.

4. IMF Web Services
The main IMF access point is through Web Services. Such an approach allows external applications to incorporate the full functionality of the IMF into various types of web and windows applications. (I-LINK is an IMF client example).
IMF web services operations provide one point access to a cluster of IMF Core functionality. The IMF web services, when called through a client, perform IMF token authentication, process the required operation and return IMF package results.
5. IMF Generic Message Services

5.1. RSS Feeds (Blogs)

RSS Feeds are very popular on the web and are being used for various types of content syndication. As such, we found it strategic to enable IMF with an RSS
'outlet'. As such IMF Message, Project and Contact information can be accessed through a structure compatible with the RSS 2.0 format.

The IMF Generic Message Services provide the capability to subscribe to existing IMF projects through a regular Blog aggregator and retrieve project messages and contacts seamlessly.

The RSS Feed classes access IMF Core classes directly and can be therefore considered as an alternative access channel to Web Services.

Figure IV-25 Project RSS Feed

```xml
<rss version="2.0">
  <channel>
    <title>test</title>
    <link>http://18.58.3.130/IMFServices/Online/Web.aspx?project=1</link>
    <description>Project Created by:td4</description>
    <language>en-us</language>
    <lastBuildDate>1/1/0001 12:00:00 AM</lastBuildDate>
    <item>
      <title>monty</title>
      <link>http://localhost/IMFServices/Online/WebContact.aspx?id=2</link>
      <guid>http://localhost/IMFServices/Online/Contact.aspx?id=2</guid>
    </item>
    <item>
      <title>yalla 4</title>
      <link>http://localhost/IMFServices/Online/WebMessage.aspx?id=12</link>
      <guid>http://localhost/IMFServices/Online/Message.aspx?id=12</guid>
      <description />
    </item>
  </channel>
</rss>
```

5.2. Email forwarding

In order to provide support for various types of clients, we have implemented basic email Project Update notifications.
Through the Mail Connector, the IMF forwarder notifies project contacts of new messages, documents, and contact additions to a project.

6. IMF Client Access

6.1. Mainstream Clients

6.1.1. Web Services Consumption
IMF functionality can be accessed from remote application through web services consumption, thus allowing the integration of its messaging capabilities within new and existing systems.

6.1.2. Web Clients
Since IMF is build as a centralized message repository. IMF project information can be accessed via an IMF portal application which provides basic framework functionality.

6.1.3. Email Clients
Through email notifications, IMF can enable any internet user with an email address to participate in complex project collaboration via IMF.

6.1.4. Blog Aggregators
IMF Projects can be access through mainstream blog aggregators which can subscribe to project feeds and stay updated on all changes.
A subscribed blog aggregator can retrieve all project information such as messages, contacts and shared documents.

6.2. I-LINK Desktop Client (Chapter V4)
The I-LINK Desktop Client is a project management client that is designed explicitly for use with the IMF framework. It makes use of most of the IMF functionality and provides a versatile IMF demo. I-LINK client accesses IMF primarily through Web Service function calls and receives event, presence and message notifications through TCP calls to its notification listeners from IMF (Figure IV-26 : IMF - I-LINK Architecture).
Figure IV-26: IMF - I-LINK Architecture

IMF ARCHITECTURE
Chapter V  I-LINK Architecture

1. What is I-LINK?
IMF is designed as an independent repository which is accessed from a variety of sources. In that respect, clients can connect to the IMF through various channels and benefit from its published functionality.

While most access channels provide relatively comprehensive access to IMF’s core features, it remains that a fully integrated client such as I-LINK presents the most complete showcase of IMF’s functionality.

I-LINK is a simple project management client that is designed explicitly for use with the IMF. It enables users to manage multiple projects by sharing correspondence, documents, and contact lists through a simple, user friendly, windows interface.

I-LINK derives its core functionality from IMF, and handles data presentation, incoming notifications and web services access through several local architectural layers.

In that respect, I-LINK can be dissected into several components that enable and supports the visual architecture: Local Data Stores, Web Services Proxy, Core I-LINK Classes, and UI Classes.

2. I-LINK Environment and Deployment Requirements

2.1. Platform
The I-LINK Desktop client is targeted for deployment in a Microsoft Windows Environment that supports the .Net Framework. I-LINK’s components are deployed as .NET executable assemblies which are managed by Microsoft’s Common Language Runtime (CLR).

As such I-LINK requires the .Net Framework to be installed on the deployment target station.

2.2. Storage Requirements
I-LINK stores message headers and temporary data in a local directory structure which it creates partially upon installation and partially as it is being used. All local data is stored in xml documents.
2.3. Deployment: Setup Wizard
The I-LINK client is composed of a main executable assembly supplemented by a set of images, help files, and directory structures. In order to insure proper deployment and a seamless user experience we packaged all I-LINK components into a user friendly setup wizard that handles all deployment steps and provides uninstall capabilities.

2.4. Access Permissions and Network Requirements
As a desktop application, I-LINK inherits the machine access permissions available to the Windows user that invokes it. As such, client installation and folder structure creation are subject to Windows access restrictions and might require Administrator privileges in secured environments.
Furthermore, the I-LINK listens to incoming TCP notifications on the machine Port 9050 which needs to be open (i.e. access allowed) on both the local machine and any enterprise firewall.

3. I-LINK Visual Architecture
Since I-LINK is aimed at a relatively varied audience, we faced early on the challenge of delivering a highly interactive yet simple client interface that would encourage users to adopt it for its ease of use and enhanced functionality.
As such we followed a design methodology that used a map of the required feature set, in conjunction with the existing IMF data structure and functionality in order to produce a modular, object centric user experience.

3.1. I-LINK User Interface
I-LINK is organized visually into three adjacent panes: Navigation Pane, Catalogue Pane and Reading Pane. These panes provide comprehensive access to all the project management components needed within the I-LINK scope: projects, contacts, messages and documents. (refer to: M.Eng Thesis, Pamela Chahine)

3.1.1. Navigation Pane
The navigation pane is the main control center for accessing I-LINK objects.

3.1.2. Catalogue Pane
The catalogue pane provides specific object lists such as messages and contact lists.

3.1.3. Reading Pane

The reading pane displays specific object information such as message information and contact details.

Figure V-1 I-LINK Main Interface

3.2. Object-Action Model

Object-Action model: The user first selects an object and then selects the action to be performed on the selected object.

Desktop users differ in the way they interact with client applications and perform tasks, therefore forcing interface designers to consider multiples paths towards the completion of a single task.
In our quest for a simple, user friendly interface we chose to adopt a task invocation path which is known as: Objects-Action model, and which we found would fit smoothly with I-LINK’s manipulation of the IMF object oriented platform. As a matter of fact, I-LINK project management objects have a direct correlation with the IMF objects/types they operate on. Ex: I-LINK project, contact and messages information is obtained through the exchange of IMF Project, IMF Contact and IMF Message objects with the IMF Web Services Tier. Operating on an I-LINK object takes place through selecting an object and choosing the task to perform.

Figure V-2  I-LINK Object-Action
4. I-LINK Architecture
I-LINK is designed as a fully object oriented client which manipulates, presents and acts on IMF objects. It is organized into four logical layers: Local Data Store, Web Services Proxy, Core Classes and UI Classes.

4.1. Local Data Stores
I-LINK retrieves its data from the IMF repository, and stores message header information and other essential data into local xml data stores therefore enabling offline access to essential information.
Furthermore, I-LINK stores local copies of full messages and documents that users chose to download from the IMF repository. The I-LINK directory structure created during the client installation provides the backbone for storing and accessing the local xml repositories and downloaded files.

4.2. IMF Web Services Proxy
Web Services are designed for inter-machine communication and is usually implemented through a client-server model. Requests by the client are handled by a local web service proxy object which forwards them to the server web service. The .Net service consumption model relies on a local assembly which is created automatically (through Visual Studio) or manually (using the wsdl.exe tool) from the Web Service's WSDL (Web Service Definition Language).
The local proxy converts native C# data types into XML messages which are transported to the web service using SOAP (Simple Object Access Protocol) and HTTP (Hypertext Transport Protocol). It also translates incoming XML messages from the Web Service Response into native C# data types.
Figure V-3 I-LINK Client- IMF Services Communication

Figure V-4 IMF Web Service Proxy
4.3. UI Classes

.Net provides a versatile framework for the development of windows applications. The I-LINK client is implemented as a Windows Application Project and the user interface is therefore made up of multiple form classes that extend the Windows Form Class.

4.3.1. .Net Windows Controls

.Net provides a rich windows development toolset which empowers user interface developers through a variety of standard windows visual controls that can be easily ‘dropped’ onto the visual studio designer platform. While designing the I-LINK Client we relied extensively on several of the above controls such as: ListView, Treeview, ComboBox, TextBox, Button, Menu, etc...

4.3.2. Login Form
The main application entry point is the login form which is a basic form class that expects authentication info.

The form submits info to the IMF Web Services processes the result, then instantiates and launches the I-LINK main form upon successful login.

4.3.3. Main Form

As in most desktop applications, I-LINK users interact with one major interface/form that acts as the application control center. The main I-LINK form is logically organized into three panes as outlined previously (Figure V-1 I-LINK Main Interface).

These panes are implemented within the Main Form Class through three Windows Controls: TreeView (Navigation Pane), ListView (Catalogue Pane) and Panel/Label/TextBox Controls (Reading Pane).
1. TreeView Control: The treeview control is the first control invoked within
the main form and is populated through multiple calls to the IMF Web
Services. The control displays the I-LINK main space and project
components through a hierarchical structure that can be acted upon by
right clicking and selecting the desired task to invoke.
Clicking on the control loads the selected object's detailed information in
the Reading Pane.

2. ListView Control: The listview control is populated from the local xml data
files and limited calls to the IMF services upon invocation of treeview
object actions. The control displays an ordered item list which can also be
acted upon by right clicking and selecting the desired task to invoke.
Clicking on the control also loads the selected object's detailed information
in the Reading Pane.

3. Panel/Label/TextBox Controls: the reading pane is composed of several
controls which are used to display detailed information about selected
objects in the catalogue or navigation pane.
The reading pane does not require any remote IMF web services calls as
it is populated by the data (IMF Type Data) encapsulated in the invoking
object. Furthermore, the reading pane allows the user to manually invoke
remote IMF calls in order to perform specific operations such as
downloading data from the IMF repository.
The main form also implements several functions that are used to populate
controls, handle user interface events and process exceptions.
4.3.4. Dialogs

While the Main Form provides the bulk of the client functionality some tasks such as composing a message, chatting, adding/modifying information and uploading documents, require their own customized interface. These are implemented as child dialogs which are invoked modally (i.e. when launched, access to the main form is blocked until the dialog is closed) from within the Main Form.

4.4. Core Classes

In addition to the essential 'Form' classes described above, the I-LINK client is based on several management classes that enable full integration with the IMF. These classes are: IMF Proxies, Data Serializers and the TCP notification Listener.
4.4.1. IMF Proxies

IMF Proxies use the class definitions provided by the IMF Web Services Proxy (Figure V-4 IMF Web Service Proxy) to provide local functionality that is consistent with the IMF Type Managers (Chapter IV3.2). As such IMF Proxies operate on local instance of IMF Object and act as the main connection between the User Interface Classes and IMF.

IMF Proxies are organized using a logical structure that is identical to that of the IMF Type Managers:

1. User Proxy: Provides local user management functionality such as creating, modifying and deleting users as well as retrieving user presence information.
2. Contact Proxy: Provides local contact management functionality such as creating, modifying, and deleting user contacts. It also provides the essential components for managing contact addresses.
3. Project Proxy: Provides local project management functionality such as creating, modifying, deleting and unsubscribing from a project. It also provides the essential components for adding project messages, contacts and documents.
4. Account Proxy: Provides local account management functionality such as creating, modifying and deleting user and project accounts.
5. Message Proxy: Provides local message management functionality such as composing, deleting and retrieving messages. It also provides extended information retrieval features such as downloading message body, processing message addresses and uploading message attachments.

4.4.2. Data Serializers

While I-LINK is essentially designed to access the IMF central repository, performance constraints have directed us toward storing essential IMF data in local data stores in order to provide offline access to essential information such as message headers and downloaded files. The I-LINK Data Serializers enable
the serialization and de-serialization of local xml data for use within the I-LINK classes. Data is retrieved and stored in the I-LINK directory structure (Chapter V4.1).

4.4.3. TCP Notification Listener

I-LINK communicates with the IMF solely through Web Service calls that are handled by the IMF Web Services Proxy (Chapter V4.2). While such a scheme is very efficient and provides a structured communication mechanism, it does not allow for Server Triggered notification, since the I-LINK client is not designed for deployment on a web server and cannot therefore receive incoming web service calls. Using the IMF Web Services Proxy, I-LINK client can only process the results of synchronous Web Services calls.

In order to cater for the more complex mechanisms required by features such as presence notification and chat sessions, we implemented a local process
that listens for and processes TCP serialized messages sent to the client’s Port 9050 from the IMF Server. Such an inbound channel enables the IMF Server to communicate with active client sessions and send notifications of IMF events such as chat requests, presence change, new messages, and project changes. The I-LINK Notification Listener process is launched upon successful login to the IMF Server and is designed to process IMF Notification Packages (Chapter IV3.1.13) submitted by the IMF Server.

Upon receiving incoming packages the Notification Listener parses the package contents, queues the resulting IMF objects onto the client’s internal thread queue, and invokes the appropriate application event handlers.

Figure V-8 TCP Notification
Chapter VI  Outcomes, Challenges and Future Work

1. Overview
As we ventured to explore the possibilities inherent in integrating multiple messaging formats, we completed several milestones which lead to the creation of an Integrated Message Framework. In the process, we came across several design challenges, development shortcomings and expansion possibilities that we identified for future pursuit. Overall, we were able to deliver a stable and powerful framework as well as versatile windows clients, which combined deliver on the majority of the tasks we had set out to accomplish.

2. IMF – I-LINK Summary
The Integrated Message Framework presented in this thesis provides a powerful platform for the integration of multiple messaging formats into a central repository. IMF is designed to leverage the features of the various media it encapsulates and as such provides an intelligent centralized platform for accessing and managing messages. Furthermore, IMF’s most important feature is its client independence which is achieved through the provision of multiple data delivery channels (RSS Feeds, Web Services, and Email Notification).
IMF, in conjunction with the I-LINK client provides a simple and versatile project management application that combines IMF’s full messaging functionality with a user friendly object centric windows interface.

3. Design Challenges
Having set out to achieve what was obviously a very ambitious goal we anticipated encountering several design challenges particularly given the compressed delivery timeframe, our limited familiarity with the development platform and the distributed nature of the application we aimed at producing:

3.1. Design Learning Curve
While we had a fairly well rounded development experience, we had not previously endeavored in any full development spiral and as such had to discover in a very
short period of time all aspects of the ‘formal’ development process, from expanding our familiarity with the .Net Platform, working with UML models to producing comprehensive application requirements. The learning curve required by the previous tasks significantly affected our production progress, yet the resulting acquired skill set significantly improved the quality of our final product.

3.2. Security Challenges

3.2.1. Resolved

Having in mind that throughout our endeavor we would face several challenges, we chose to sideline most security considerations and focus on the theoretical development of the framework. Nevertheless, as we went on to execute our plans we were faced with security obstacles that blocked our efforts and required resolution such as:

1. Securing Web Services Access: as web services are published through the Internet, it was essential for us to secure their access in order not compromise the security of the server on which they are deployed. As such we implemented a token checking mechanism that provides an additional web services security layer and limits the possibility of malicious requests being invoked.

2. Securing Database Access: while all database requests are invoked from private IMF Types Managers, we still had to ensure that the publicly invoked ASPNET account (web services account) which controls the IMF Type Managers does not allow for malicious access to the database. We therefore implemented a non Windows SQL Server Account that is intended solely for db access and that is not controlled by the ASPNET process.

3. Securing Password Exchange: As IMF data types are exchanged through SOAP xml messages, we had to encode user passwords in a manner that prevents malicious access to the info should the message be intercepted. As such we resorted to exchanging and storing passwords after producing a unidirectional hash code derived from the initial password string.
3.2.2. Outstanding

1. Secure Communication: current IMF communication is solely through text based SOAP xml messages. Such formats are not encrypted and should they be intercepted would present a significant security issue. Future framework implementations should support encrypted exchanges in order to scale for usage in enterprise communication.

2. Serialization Security: as data is exchanged in the form of serialized IMF data types, a malicious user could attempt to submit properly formatted data types containing malicious instructions. The current IMF implementation does not validate data types beyond the initial token validation and as such cannot prevent such 'intrusions'. Future framework implementations should tackle this issue through either encrypted data exchanges or proprietary internal data validation.

3.3. Web Services Challenges

1. Web Service Access: as web services are deployed on the internet yet are invoked in the same manner as local procedures, we were tempted early in our development of the I-LINK client to treat them as local function calls. Such an approach proved to be quite problematic as multiple calls significantly slowed the local application and raised multiple client exceptions.

   In order to resolve that matter we had to review our web services approach and limit web services requests to the bare essentials required for seamless operation while maintaining several local data buffers/stores which ensure smooth user interaction.

2. Service Description: while the WSDL (Web Service Description Language) is very useful, it has not yet reached its full maturity level and as such function overrides and multiple class support cannot be implemented within one web service. Such a limitation has forced us to use complex web service function names that do not closely map the IMF Type Managers.
4. Development Shortcomings
While we were able to deliver a majority of the IMF-l-LINK functionality we had set out to product, there were nevertheless several development shortcomings:

4.1. Comprehensive support for existing Media
Although, IMF is designed to work with most existing messaging media, its current version of IMF does not provide active support for all existing media. As such, future implementations should provide support for: IMAP, Consumer IM (Yahoo, MSN, etc...), and RSS versions prior to the 2.0 standard.

4.2. Full Calendar Event and Task Support
The IMF data structure, through the message rule data types provides support for future calendar and event implementations. Nevertheless, given the project’s limited timeframe, we were unable to implement the calendaring functionality within the current IMF release.

4.3. Support for all Mainstream IM Platforms
The current IMF implementation provides support for IMF system IM. Support for mainstream IM platforms requires the development of individual platform translators that would handle message parsing. Future framework implementations should support consumer IM as well as corporate IM platforms.

5. Framework Expansion Possibilities
IMF’s integration of multiple messaging formats provides an interesting platform for experimenting with different message applications, as such we identified several approaches that would be worthwhile for further development:

5.1. Intelligent Message Parsing
As IMF processes messages from several sources through one central repository. We believe that it would be very interesting to introduce intelligent processing agents that can parse specific message embedded tags and deliver ‘intelligent’ content to the message’s recipients.
5.2. Message Registration

In addition to its explicit functionality, the IMF can act as a 'validation' repository for various media. As such a client application can submit message registration to the IMF, obtain a registration ID (from the IMF) and dispatch the message to its intended recipients. In that scenario the IMF acts as an intermediary registration authority which validates and confirms the message’s author.

5.3. Inclusion of Legacy Message Media

The IMF can be extended to provide support for legacy message media such as Faxes. As such, future implementations would be able to integrate fully within environments that are not 'internet enabled'.
Bibliography

Appendix A  IMF Class Packages

IMFServices::MediaParsers

IMFServices::IMF Types

IMFServices::Media Connectors

Database Interface

Type Managers

Top Package::.NET System

IMFServices::Generic Message Services

IMFServices::Web_Services
Appendix B  Selected Architecture Sequences

Message Package Parsing Sequence

<table>
<thead>
<tr>
<th>WS</th>
<th>IME Pr</th>
<th>Session</th>
<th>User</th>
<th>Presence</th>
<th>Project</th>
<th>Message</th>
<th>Contact</th>
<th>Address</th>
<th>Account</th>
<th>Connectors</th>
<th>DB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- [User] Check Presence
- Add Message
- Insert Message Reader
- Return Message ID
- Insert Message Parts
- Insert Message Contacts
- Insert Message Addresses
- Insert Message Rules

Type Managers

93