Windows Eudora IMAP Architecture
February 1, 2006

1 Introduction to the IMAP Projects

Eudora's IMAP functionality is broken down into two separate projects: Imap and EuImap. The Imap project implements communication with an IMAP server and the EuImap project ties the Imap project into Eudora. EuImap is the only project that calls directly into the Imap project.

[INSERT DIAGRAM OF PROJECT RELATIONSHIPS]

2 The Imap Project

The Imap project implements the communication between the client and the IMAP server. Although it uses two other projects within the Eudora workspace (QCSocket and QCUtils) the Imap project knows nothing about Eudora specific data and has no access to the INI file. The code in this project is compiled into a separate DLL named Imap.dll.

2.1 Fundamental Classes

The following classes are fundamental to understanding the Imap project code:

CNetStream – Contains the functionality for managing network connections.

CStream – Contains various state info and error handling functions related to the IMAP stream.

CProtocol – Formats commands to be sent to the IMAP server and parses the data that is returned. This class is derived from both the CNetStream and CStream classes and in addition to the inherited functionality it implements methods for issuing all IMAP commands to the server and parsing the data returned by the server. This class uses a mailgets_t callback function to write the returned data to a CWriter object.

CImapStream – Provides the interface into the Imap project. The EuImap project uses this class for all interaction with the Imap project. This class creates a CProtocol object and calls into it to perform IMAP operations.

CWriter – Abstract base class for classes that write IMAP data to an unspecified destination. There are three subclasses (which are defined in the EuImap project):

CStringWriter which writes the data to a string, CFileWriter which writes the data to a file and CImapDownloader which writes the data to an individual message file.

CReader – Abstract base class for classes that read IMAP data from an unspecified source. Subclasses implement reading from a specific source. Currently the only subclass is CFileReader which is used only by the CImapAppend class to read the file containing the message data to be appended to the server.

CLister – Abstract base class for classes that manage lists of IMAP data. There are two subclasses: CMboxLister which contains a list of mailboxes as returned by the server and CImapNodeLister which contains attribute data returned by the server for a list of mailboxes.

2.2 Sockets

There is a one-to-one correspondence between CImapStream objects and CProtocol objects. Each CImapStream creates and maintains one CProtocol object for its lifetime. Likewise each CProtocol object creates a new socket to the IMAP server and keeps that socket open for its lifetime.

2.3 Communicating with EuImap

The CImapStream object (in the Imap project) communicates success or failure back to the CImapConnection object (in the EuImap project) via a return code. On failure, the CImapStream will cache the error and the CImapConnection object can obtain the error message via the CImapStream::GetLastErrorStringAndType() method.

In most cases, the IMAP server will return data other than the result code (success or failure). This is handled by providing the CImapConnection object with an instance of a subclass of the CWriter object. CWriter objects serve to write the data returned by the server to a specific location, for example to a string or to a file. Code in the EuImap project can then access the data from the string or file.

3 The EuImap Project

The EuImap project provides the interaction between Eudora and the Imap project. This project has access to the core Eudora code. It calls into the Imap project to issue commands to the IMAP server and processes the data returned by the server via the Imap project. The code in this project is compiled into the Eudora.exe file.

3.1 Fundamental Classes

The following classes are fundamental to understanding the EuImap project code:

CImapAccount – Represents a personality that uses IMAP.

CImapAccountList – Array of CImapAccount objects. This array contains one CImapAccount object for each account whose data is stored in a given directory.

CImapAccountMgr – Manages the entire list of IMAP accounts. It contains a list of CImapAccountList objects each representing a different data directory. A single global instance of this class is created and used to manage the IMAP accounts.

CImapMailbox – Object for managing the mailboxes associated with a given CImapAccount object. Includes UID Validity and flags associated with the mailbox on the server. Maintains a pointer to the corresponding CTocDoc object.

CTocDoc – There is no subclass of CTocDoc for IMAP TOC's, but rather certain methods and data that apply only to IMAP mailboxes. These portions of CTocDoc are implemented in the EuImap project. Maintains a pointer to the corresponding CImapMailbox object.

CImapMailMgr – Manages requests to resync mailboxes. A single global instance of this object exists and manages all mail checks for all IMAP accounts.

CImapChecker – Implements the functionality of resyncing a local mailbox to reflect the contents of the mailbox on the server.

CImapDownloader – Implements the functionality of downloading one or more parts of a message.

CImapAppend – Implements the functionality of appending a new message to an IMAP mailbox.

CImapConnection – Encapsulates IMAP commands. Handles mutex locking and calls into the Imap project to issue IMAP commands to the server. Any class that needs a connection to the IMAP server should create an instance of this class and call the appropriate method.

CImapConnectionMgr – Manages a list of CImapConnection objects. A single global instance of this object exists and manages all connection objects for all accounts. When a connection is desired the calling code asks the global connection manager for a connection. The connection manager looks to see if a connection already exists for the given account and, if specified, mailbox. If no such connection exists the manager will create one. If a connection's refcount indicates a connection is no longer being used then the connection object will be deleted.

CActionQueue – Implements a queue of IMAP actions to be executed. See the “IMAP Action Queuing” section for further details about the action queue.

CImapAction – Abstract base class for IMAP actions that are to be executed at a later time. Each kind of IMAP action has a subclass of this class which performs the appropriate action online.

3.2 Sockets

There is a one-to-one correspondence between CImapConnection objects in the EuImap project and CImapStream objects in the Imap project. When a connection is requested, the CImapConnection object creates and maintains a CImapStream object.

4 IMAP Action Architecture

Starting with version 7.0 Eudora executes IMAP commands via an action queue. The primary motivation for the action queue is to allow actions to be queued while Eudora is offline and replayed while online. This same architecture is also used while Eudora is online which not only simplifies the code but also allows for performance improvements. Details of the action queue and offline mode are discussed in the “IMAP Action Queue” and “Offline IMAP” documents respectively.

5 Code Flow

The code flow can be a little convoluted, especially with the addition of the action queue where multiple threads might be involved in executing a single action.

5.1 Flow Diagram

The following diagram shows the interaction between objects in the Imap and EuImap projects.

[INSERT DIAGRAM OF CLASS INTERACTIONS]

5.2 Sample Call Stack

The following call stack illustrates the process of syncing a mailbox. The first section starts by showing the mailbox tree control processing the user request to check mail for a given IMAP mailbox and ends with the creation of a CImapResyncAction action object which is then added to the action queue:

```
CMboxTreeCtrl::OnCmdImapResync()
    QCImapMailboxCommand::Execute()
        CImapMailbox::DoManualResync()
            CImapMailbox::DoBackgroundResync()
                GetMail()
                    CImapMailMgr::ImapGetMail()
                        CImapMailMgr::CheckMailInBackground()
                            CImapMailMgr::CheckMail()
                                CImapResyncAction::CImapResyncAction()
                                CImapAccount::QueueAction()
```

When the CImapResyncAction completes it calls back into the CImapMailMgr class to perform a mail check. This results in the creation of a CImapChecker object which also executes in its own thread:

```
CImapResyncAction::DoPostProcessing()
    CImapMailMgr::CheckMailOnServer()
        CImapChecker::CImapChecker()
```

Finally, the CImapChecker object is executed in its own thread, creating a CImapConnection object which calls into the Imap DLL via CImapStream which uses a CProtocol object to communicate with the server:

```
CImapChecker::DoWork()
    CImapChecker::DoCheckMailMT()
        CImapChecker::GetNewMboxState()
            CImapConnection::Noop()
                CImapStream::Noop()
                    CProtocol::Ping()
```