(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-502239
(P2004-502239A)
(43)公表日　平成16年1月22日(2004.1.22)

(51)Int.Cl.[7]	F I	テーマコード（参考）
G06F 13/00	G06F 13/00 620	

審査請求 未請求　予備審査請求 有　（全 128 頁）

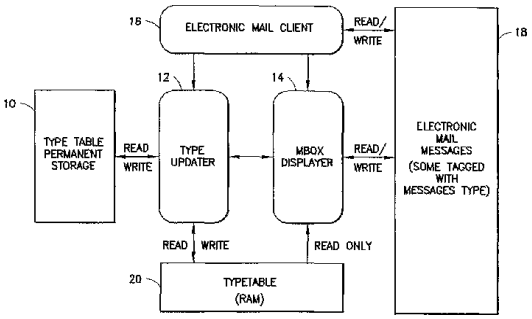
(21)出願番号	特願2002-506441(P2002-506441)
(86)(22)出願日	平成13年6月20日(2001.6.20)
(85)翻訳文提出日	平成14年12月27日(2002.12.27)
(86)国際出願番号	PCT/US2001/020348
(87)国際公開番号	WO2002/001373
(87)国際公開日	平成14年1月3日(2002.1.3)
(31)優先権主張番号	09/604,426
(32)優先日	平成12年6月27日(2000.6.27)
(33)優先権主張国	米国(US)

(71)出願人　503012384
インテリネット　インコーポレイティド
アメリカ合衆国，ニューヨーク　１００２５，ニューヨーク，リバーサイド　ドライブ　３２５
(74)代理人　100077517
弁理士　石田　敬
(74)代理人　100092624
弁理士　鶴田　準一
(74)代理人　100122965
弁理士　水谷　好男
(74)代理人　100082898
弁理士　西山　雅也
(74)代理人　100081330
弁理士　樋口　外治

最終頁に続く

(54)【発明の名称】ＭＩＭＥタイプを識別し異なるアイコンを表示する方法及び装置を含む機能拡張された電子メールシステム

(57)【要約】
電子メールクライアントソフトウェア（１６）は、各メッセージ用のアイコンと共にメッセージを一覧表示するメールボックスディスプレイヤ（１４）を備えており、これらのアイコンは各メッセージのＭＩＭＥタイプに関連付けられている。添付ファイル（１８）を含むメールは、そのメールに添付されたファイルのタイプを示すアイコンと共に受信ボックスに一覧表示される。メールボックスディスプレイヤがＭＩＭＥタイプを解釈し、ＯＳ内のアイコンレジストリ又はＥメールクライアントソフトウェアが維持管理するアイコンディレクトリから適切なアイコンを選択する。アイコンディレクトリに適切なアイコンが存在しない場合には、メールボックスディスプレイヤは、ＭＩＭＥメッセージのサブパートに含まれているアイコン画像データを使用する（データが提供されている場合）。そうでなければ、アイコンを使用しないか、或いは一般アイコンが使用される。好適な実施例によれば、タイプテーブル（２０）がタイプアップデータ（１２）コンポーネントによって維持管理される。このタイプテーブルには、メールボックスディスプレイヤ



【特許請求の範囲】
【請求項１】
表示装置と共に使用する電子メールクライアントソフトウェアであって、
ａ）メールボックスの内容を前記表示装置上にスクロール可能なリストとして表示するメールボックスディスプレイヤ手段であって、各メールアイテムが複数のプロパティと共に一覧表示され、前記プロパティは件名、送信者の名前、及び送信日付からなる群から選択されるメールボックスディスプレイヤ手段と、
ｂ）複数のＭＩＭＥタイプを複数のアイコン画像と関連付ける関連付け手段と、
を有し、
前記メールボックスディスプレイヤ手段は、メールボックス内の少なくともいくつかのメールアイテムのＭＩＭＥタイプを判定する手段と、前記関連付け手段を読み取る手段と、前記スクロール可能なリスト内にアイコン画像を前記少なくともいつかのメールアイテムのそれぞれに関連付けられたプロパティとして表示する手段と、を含み、
前記少なくともいくつかのメールアイテムのそれぞれ用のアイコン画像は、前記少なくともいくつかのメールアイテムのそれぞれのＭＩＭＥタイプに従って前記関連付け手段から選択されることを特徴とする電子メールクライアントソフトウェア。
【請求項２】
ｃ）前記関連付け手段を更新して更なるＭＩＭＥタイプと更なるアイコン画像を含めるタイプアップデータ手段を、
更に有する請求項１記載の電子メールクライアントソフトウェア。
【請求項３】
ｃ）それぞれがファイル名を備える複数のアイコン画像を更に有し、前記関連付け手段は、少なくともいくつかのＭＩＭＥタイプのそれぞれを前記アイコン画像のファイル名と関連付けるデータ構造である請求項１記載の電子メールクライアントソフトウェア。
【請求項４】
前記関連付け手段は、ＭＩＭＥタイプをプログラムと関連付ける手段を含み、いくつかのＭＩＭＥタイプは、アイコン画像とは関連付けられず、プログラムと関連付けられる請求項１記載の電子メールクライアントソフトウェア。
【請求項５】
前記タイプアップデータ手段は、ユーザー入力に応答する請求項２記載の電子メールクライアントソフトウェア。
【請求項６】
前記複数の画像の少なくともいくつかは拡大縮小可能である請求項３記載の電子メールクライアントソフトウェア。
【請求項７】
前記電子メールクライアントソフトウェアは、アイコンのレジストリを維持管理するオペレーティングシステムと共に使用するように設計されており、
前記関連付け手段は、少なくともいくつかのＭＩＭＥタイプを前記アイコンのレジストリから選択されたアイコンと関連付ける請求項１記載の電子メールクライアントソフトウェア。
【請求項８】
ｃ）Ｅメール又はＥメールの添付ファイルに格納されているグラフィカルアイコン情報を読み取るアイコン回復手段を更に有し、
前記関連付け手段は、その他のアイコンが見つからない場合に、ＭＩＭＥタイプを前記アイコン回復手段によって回復されたアイコンと関連付ける請求項１記載の電子メールクライアントソフトウェア。
【請求項９】
前記タイプアップデータ手段は、前記電子メールクライアントソフトウェアのユーザーが手動で操作可能である請求項２記載の電子メールクライアントソフトウェア。
【請求項１０】

前記タイプアップデータ手段は、
新しいＭＩＭＥタイプのオーサリング／読み取りを行う能力を有する新しいアプリケーション又はコンポーネントがインストールされるイベントと、
受信したメールメッセージにおいて新しいＭＩＭＥタイプを検出するイベントと、
定期的にスケジュールされたイベントにより新しいアイコンを取得するべく前記タイプアップデータがサーバーをチェックするイベントと、
から構成される群から選択されたイベントに従って新しいアイコンを自動的にインストールする請求項２記載の電子メールクライアントソフトウェア。
【請求項１１】
電子メールボックスの内容リストを表示装置上に表示する方法であって、
ａ）前記メールボックスの内容をスクロール可能なリストとして前記表示装置上に表示する段階であって、各メールアイテムは複数のプロパティと共に一覧表示され、前記プロパティは、件名、送信者の名前、送信日付からなる群から選択される段階と、
ｂ）前記メールボックス内の少なくともいくつかのメールアイテムのＭＩＭＥタイプを判定する段階と、
ｃ）前記スクロール可能なリスト内にアイコン画像を前記少なくともいくつかのメールアイテムのそれぞれと関連付けられたプロパティとして表示する段階であって、前記少なくともいくつかのメールアイテムのそれぞれ用のアイコン画像は、前記少なくともいくつかのメールアイテムのそれぞれのＭＩＭＥタイプに従って選択される段階と、
を有することを特徴とする方法。
【請求項１２】
ｄ）ＭＩＭＥタイプをプログラムと関連付ける段階と、
ｅ）あるＭＩＭＥタイプのメールアイテムが開かれた際に当該ＭＩＭＥタイプと関連付けられたプログラムを実行する段階と、
を更に有する請求項１１記載の方法。
【請求項１３】
前記アイコン画像の少なくともいくつかは拡大縮小可能である請求項１１記載の方法。
【請求項１４】
前記表示段階は、アイコンの中央レジストリから少なくともいくつかのアイコンを選択する段階を含む請求項１１記載の方法。
【請求項１５】
前記表示段階は、Ｅメール又はＥメールの添付ファイルに格納されたグラフィカルアイコン情報を読み取る段階を含む請求項１１記載の方法。
【請求項１６】
ｄ）前記表示段階を実行する際に使用するグラフィカルアイコンの保存データを維持管理する段階を、
更に有する請求項１１記載の方法。
【請求項１７】
ｅ）新しいＭＩＭＥタイプのオーサリング／読み取りを実行する能力を有する新しいアプリケーション又はコンポーネントがインストールされた際と、
受信したメールメッセージ内で新しいＭＩＭＥタイプを検出した際と、
定期的にスケジュールされたイベントにより新しいアイコンを取得するべく前記タイプアップデータがサーバーをチェックした際と、
からなる群から選択されたイベントに応答して前記グラフィカルアイコンの保存データ内に新しいアイコン画像を自動的にインストールする段階を、
更に有する請求項１６記載の方法。
【発明の詳細な説明】
【０００１】
（関連出願の相互参照）
本出願は、１９９８年１２月１０日付けの米国特許出願第０９／２０９，１６２号の一部

継続出願であり、本引用により、その開示内容のすべてが本明細書に包含される。又、本出願は、同時に出願された同時継続中の米国特許出願第ＢＡＫ－００７号と関連しており、そのすべての開示内容も本引用によって本明細書に包含される。

【０００２】

（発明の分野）

本発明は電子メールプログラムに関し、更に詳しくはメールアイテムのＭＩＭＥタイプごとに異なるアイコンを表示するメールボックスブラウザ表示を備えた電子メールプログラムに関する。

【０００３】

（本技術分野の状況）

近年、電子メール（「Ｅメール」）が、ビジネス、教育、及び個人的な通信に広く使用されるようになっている。電子メールの（特に、ビジネス及び教育分野における）最も便利な機能の１つがバイナリコンピュータファイルをＥメールメッセージに添付する能力である。この機能により、Ｅメールの利用者は、ワードプロセッサ文書、データベース文書、表計算文書、マルチメディア文書などのコンピュータによって作成されたほとんどすべての種類のバイナリファイルを迅速に共有することができる。しかしながら、Ｅメールメッセージへのバイナリファイルの添付には深刻な制約と不都合が存在している。

【０００４】

ＲＦＣ（Ｒｅｑｕｅｓｔ ｆｏｒ Ｃｏｍｍｅｎｔｓ）８２１及び８２２によって１９８２年に定義されたオリジナルのインターネットメールシステムには、いくつかの重要な制約事項が存在していた。特に、このシステムは、Ｅメールメッセージで任意の大量データを搬送するように設計されていなかったのである。現実に、１９８２年のＳＭＴＰ（Ｓｉｍｐｌｅ Ｍａｉｌ Ｔｒａｎｓｐｏｒｔ Ｐｒｏｔｏｃｏｌ）規格によれば、Ｅメールメッセージは、行当たりの文字数が１０００文字（３２ｋのブロック）以下のＡＳＣＩＩ文字のみからなる単一メッセージで構成する必要があった。

【０００５】

インターネット電子メールシステムを介してバイナリデータを送信する能力は、インターネットメッセージ用のＭＩＭＥ（Ｍｕｌｔｉｐｕｒｐｏｓｅ Ｉｎｔｅｒｎｅｔ Ｍａｉｌ Ｅｘｔｅｎｓｉｏｎｓ）規格によって実現されたものである。オリジナルのＭＩＭＥ規格は、インターネットＲＦＣ文書（ＲＦＣ１３４１）として公表され、１９９２年６月に承認されている（最新のＭＩＭＥ規格については、ＲＦＣ２０４５、２０４６、及び２０４７を参照されたい）。このＭＩＭＥ規格には、ＭＩＭＥに準拠するためのＥメールのフォーマット方法が記述されている。ＭＩＭＥには、１組のメッセージヘッダフィールドと１組のメッセージ符号化規格が定義されており、これらは、ＲＦＣ８２２のメッセージフォーマットに存在していた制約を克服すると共にインターネット上で使用されている多数のレガシーメール配送システムのいずれを介しても配送できるように設計されている（具体的には、ネットワークワーキンググループ（Ｎｅｔｗｏｒｋ Ｗｏｒｋｉｎｇ Ｇｒｏｕｐ）のＮ．フリード（Ｎ． Ｆｒｅｅｄ）及びＮ．ボレンスタイン（Ｎ． Ｂｏｒｅｎｓｔｅｉｎ）による１９９６年１１月発行のＲＦＣ２０４５「ＭＩＭＥの第１部：メッセージ本文のフォーマット（Ｍｕｌｔｉｐｕｒｐｏｓｅ Ｉｎｔｅｒｎｅｔ Ｍａｉｌ Ｅｘｔｅｎｓｉｏｎｓ （ＭＩＭＥ） Ｐａｒｔ Ｉ： Ｆｏｒｍａｔ ｏｆ Ｍｅｓｓａｇｅ Ｂｏｄｉｅｓ）」を参照されたい）。ＭＩＭＥメッセージヘッダフィールドでは、ＲＦＣ８２２に定義されていたものを拡張し、Ｅメールメッセージの内容と符号化タイプを記述する。ＭＩＭＥ規格で許されている符号化体系には、「ｑｕｏｔｅｄ－ｐｒｉｎｔａｂｌｅ」と「ｂａｓｅ６４」が含まれている。更に、これに加えて、３つの非符号化データタイプも許されており、これらは、「８ｂｉｔ」、「７ｂｉｔ」、及び「ｂｉｎａｒｙ」と呼ばれている。尚、レガシーゲートウェイでは、まだバイナリデータを処理しておらず、ほとんどすべてのＭＩＭＥ準拠メッセージは、バイナリデータをＭＩＭＥの既定の符号化体系である「７ｂｉｔ」として処理していることに留意されたい。

【０００６】

現在、ＭＩＭＥは、主要な電子メールクライアント、即ち、「ユーザーエージェント」のすべて（例：Ｍｉｃｒｏｓｏｆｔ Ｏｕｔｌｏｏｋ及びＯｕｔｌｏｏｋ Ｅｘｐｒｅｓｓ、Ｎｅｔｓｃａｐｅ Ｃｏｍｍｕｎｉｃａｔｏｒ、及びＱｕａｌｃｏｍｍ Ｅｕｄｏｒａ）において実装されている。但し、これらのプログラムが処理できるのは、「ｔｅｘｔ／ｐｌａｉｎ」、「ｔｅｘｔ／ｈｔｍｌ」、「ｍｕｌｔｉｐａｒｔ／ａｌｔｅｒｎａｔｉｖｅ」、及び「ｍｕｌｔｉｐａｒｔ／ｍｉｘｅｄ」を含むいくつかのＭＩＭＥタイプのみである。恐らく、ＭＩＭＥ規格の最も重要な機能は、あらゆるバイナリデータを適切に符号化し古いメールゲートウェイ及びエクスチェンジのＳＭＴＰシステムを介して送信することを可能にしたということであろう。前述のものなどの各メールクライアントプログラムは、あらゆるタイプのファイルをユーザーがメールメッセージに添付できるように変更されており、これは、（ａ）任意ファイルのバイナリデータを「７ｂｉｔ」や「ｂａｓｅ６４」などの認められているＭＩＭＥ符号化体系に変換する符号化モジュールの組み込み、（ｂ）ＭＩＭＥタイプが「ｍｕｌｔｉｐａｒｔ」に設定されているメッセージを処理するためのメールクライアント機能の拡張、（ｃ）ユーザーが指定したファイルを「ｍｕｌｔｉｐａｒｔ」メッセージの一部として組み込むこと、によって実現されたものである。従って、長年にわたってメールクライアントプログラムがユーザーに提供してきたのは、単純なテキストメッセージの送信（「ｃｏｎｔｅｎｔ－ｔｙｐｅ＝ｔｅｘｔ／ｐｌａｉｎ」による送信）と、単純なテキストメッセージへのファイルの添付（「ｃｏｎｔｅｎｔ－ｔｙｐｅ＝ｍｕｌｔｉｐａｒｔ／ｍｉｘｅｄ」）という２つの選択肢のみであったのである。

【０００７】

最近になり、前述の各プログラムは、作成者が別のフォントやスタイルなどのテキストフォーマットの基本的なタイプを使用できるように拡張されているが、これは、これらの機能をメールクライアントのテキストエディタに組み込み、ＭＩＭＥタイプを「ｔｅｘｔ／ｈｔｍｌ」に設定してメッセージを送信することによって実現されている。現在、マイクロソフト（Ｍｉｃｒｏｓｏｆｔ）社のＯｕｔｌｏｏｋの場合には、フル機能テキストエディタであるＷｏｒｄによる電子メールメッセージの作成も可能になっており、これは、Ｗｏｒｄのファイルフォーマットを ＨＴＭＬに変換した後に、メールメッセージの本文に手作業で挿入して送信することによって実現されている。しかしながら、その他のタイプのファイルフォーマットの場合には、メールクライアントプログラムは、まだＭＩＭＥタイプを「ｍｕｌｔｉｐａｒｔ」に設定したメッセージへのファイルの添付に依存しているのである。

【０００８】

バイナリファイルが添付されたＥメールメッセージの送信者と受信者がブランドとバージョンが同一のＥメールプログラムを使用していると共に、両方のプログラムの設定が実質的に同一であれば、受信者のＥメールプログラムは、添付されたバイナリファイルに適切な復号を自動的に施し、送信者がＥメールに添付したファイルとまったく同一のファイルを生成するであろう。しかしながら、送信者と受信者が使用するＥメールプログラムが異なる場合には、受信したファイルを復号するのに別の復号プログラムの使用を要することもある。

【０００９】

ファイルが適切に受信並びに復号された後においても、当該ファイルを開くことがファイルの受信者にとって困難な場合も多い。ファイルの受信者は「ファイルのアイコンをクリックすればファイルが開く」と考えるのだが、ファイルのアイコンをクリックしてもファイルが開かないことも多く、「ａｐｐｌｉｃａｔｉｏｎ ｎｏｔ ｆｏｕｎｄ」のようなエラーメッセージが表示されたり、ひどい場合には、不適切なアプリケーションによってファイルが開かれ、「意味不明な文字列」が表示されたりするのである。即ち、ファイルの受信者は、ファイルを読み取る（開く）ことができるプログラムを備えていなければならないのであり、例えば、Ｅメールメッセージに表計算ファイルが添付されておれば、当該ファイルをファイルの受信者が開くためには、表計算プログラムを備えていなければな

らない。無論、技術的には、ファイルの受信者は当該ファイルを作成したものと同一ブランドのプログラムを備えている必要はない。しかしながら、作成したものとは異なるプログラムによってファイルを開くのは可能だとはいうものの、それはかなり厄介なことになり得る。即ち、ファイルの受信者は、どのような種類のファイルがＥメールメッセージに添付されているのかを知らなければならず、自分のコンピュータに存在するどのプログラムがそのファイルタイプを読み取ることができるのかを知らなければならず、そのプログラムを起動しなければならず、そのプログラム内から当該ファイルを開かねばならず、更には、そのプログラムが当該ファイルを変換する間、待機しなければならないのである。

【００１０】

このようなインターネット電子メールに伴う制約事項は、送信者と受取人が使用するオペレーティングシステム（ＯＳ）が異なる場合には、更に厄介なものになり得る。メール添付ファイルの符号化体系（及びファイル圧縮体系）の中にはＯＳに依存したものがあり、受信したファイルを復号（又は解凍）できないということも起こり得るのである。

【００１１】

電子メールに伴うこれらの制約事項により、ファイルの電子メールメッセージへの添付操作は、多くの人々（特に上級者ではないコンピュータユーザー）にとって億劫なものになっている。実際、初心者ユーザーの中には、１つのアプリケーションを起動して文書を作成し、その文書を保存し、別のＥメールアプリケーションを起動してＥメールメッセージを作成し、先程保存しておいた文書を探し出してＥメールメッセージに添付するという作業が余りに難しく、怯んでしまう者もいるのである。更に、初心者ユーザーの場合には、Ｅメールメッセージに添付されたファイルを「ダウンロード」した後に、当該ファイルをハードディスク上で見つけ出すことができないという不満を漏らす者も多い。

【００１２】

【外１】

大部分のＥメールクライアントソフトウェアでは、特定のメールアイテムをユーザーが簡単に見つけ出せるよう、受信ボックス内のアイテムを送信者、件名、又は日付によって並べ替えることができる。又、大部分のＥメールクライアントソフトウェアでは、メッセージに添付ファイルが含まれているかどうかを表示しており、これは、例えば、ペーパークリップアイコンや一般文書アイコン又はフロッピーディスクアイコンなどのアイコンで示される。しかしながら、添付ファイルの特性には無関係に同一のアイコンが使用されており、当該メッセージを開くまで添付ファイルの特性を知る方法は存在していない。図１には、従来技術による代表的なＥメールの受信ボックスが示されており、いくつかのメールアイテムの件名の左隣にペーパークリップアイコンが付加され、添付ファイルの存在を示している。尚、この図１には具体的に示されていないが、当業者であれば、！、ａ、Ă、４などの一般アイコンをメッセージの件名のそばに表示し、それが優先順位の高いメッセージであるかどうか、そのメッセージに既に返信しているかどうか等のメッセージの様々な「プロパティ（属性）」を表示可能であることを理解するであろう。これらの一般アイコンは、普通、「ディングバット（ｄｉｎｇｂａｔｓ）」フォントなどから採取された単色フォント文字である。

【００１３】
【外２】

主要なＥメールクライアントプログラムの最新バージョンでは、ユーザーがメッセージを開くと、添付ファイルのタイプを表すアイコンがメールメッセージの本文に表示される。これが可能なのは、Ｍｉｃｒｏｓｏｆｔ　ＷｉｎｄｏｗｓやＭａｃｉｎｔｏｓｈ　ＯＳなどのコンピュータオペレーティングシステムが、システムにとって既知の各ファイルタイプと情報を関連付けるデータを維持管理しているためである。この情報には、グラフィカルアイコンと、当該ファイルの「オープン」や「編集」などのいくつかの操作を実行するのに使用可能なプログラムの場所が含まれている。例えば、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓでは、システムにとって既知の各ファイルタイプのエントリがシステムレジストリに含まれており、前述の情報の少なくとも一部がそのファイルタイプと関連付けられている。そして、ユーザーが「ｃｏｎｔｅｎｔ－ｔｙｐｅ＝ｍｕｌｔｉｐａｒｔ／ｍｉｘｅｄ」の電子メールメッセージを開くと、Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓ用に開発されたメールクライアントプログラム（例：Ｍｉｃｒｏｓｏｆｔ　Ｏｕｔｌｏｏｋ）は、メッセージの第２部分が添付ファイルであると判定し、「Ａｔｔａｃｈｍｅｎｔ　Ｃｏｎｖｅｒｔｅｄ：　“ｃ：＼ａｔｔａｃｈ＼ａＦｉｌｅ．ｄｏｃ”」などのメッセージ内のテキスト行を識別し、システムレジストリをチェックしてファイルタイプ「．ｄｏｃ」に関連付けられたアイコンを取得し、当該メッセージの本文にそのグラフィカルアイコンを表示するのである。

【００１４】
現在のシステムでは、アイコンをファイルに関連付けるのに、ＭＩＭＥタイプではなく、ファイルタイプ拡張子が使用されている。この結果、バージョンが異なるソフトウェアによって作成されたバージョンの異なるソフトウェアや文書を異なるアイコンと関連付ける機能に重要な制約が生じている。例えば、ＭＳ　Ｗｏｒｄで作成されたすべての文書には、使用したＷｏｒｄのバージョンとは無関係に同一のファイルタイプ（ファイル拡張子）が付加され、この結果、同一のアイコンと関連付けられてしまう。しかしながら、多くの場合、新しいバージョンのファイルは古いバージョンのソフトウェアで読み取ることができないため、問題となる。

【００１５】
先程包含された親出願には、メインＥメールコンポーネントといくつかのインストール可能コンポーネントを含む電子メールソフトウェアが開示されている。インストール可能コ

ンポーネントには、様々な種類の文書の作成／読み取りを行うオーサリング／読み取りコンポーネント（ａｕｔｈｏｒｉｎｇ／ｒｅａｄｉｎｇ　ｃｏｍｐｏｎｅｎｔ）と様々な種類のメッセージを一覧表示したり異なるスタイルでメッセージを一覧表示するメールボックスコンポーネントが含まれている。メインＥメールコンポーネントは、電子メールメッセージの保存と転送に直接関連する機能の基礎となるグラフィカル・ユーザー・インターフェイスを提供すると共に、オーサリング・コンポーネントで作成されたメッセージをＭＩＭＥに準拠したメッセージに変換するのに必要なすべてのデータのバンドリングとアンバンドリングを処理する。オーサリング／読み取りコンポーネントは、Ｅメールプログラムに組み込まれたアプリケーションのように機能し、表計算、グラフィック、データベースなどの特定のタイプの文書をＥメールプログラム内から作成して直接Ｅメールすることを可能にする。又、オーサリング／読み取りコンポーネントは、Ｅメールレターへのバイナリファイルの添付に伴う従来の問題なしに受信文書を読み取ることを可能にしている。本発明のオーサリングコンポーネントがデータをメインＥメールコンポーネントに渡し、メインＥメールコンポーネントがデータをＭＩＭＥに準拠したメッセージとしてパッケージ化する。一方、メッセージを受信した場合には、メインＥメールコンポーネントがＭＩＭＥメッセージを（必要に応じて）連結して復号し、そのデータがＭＩＭＥタイプと関連付けられたオーサリング／読み取りコンポーネントに送られる。

【００１６】

先程包含された親出願には、メールボックス処理ソフトウェアが広範に開示並びに権利請求されており、このソフトウェアにより、モジュラーコンポーネントＥメールソフトウェアのコンテキストにおいて、様々なタイプのメッセージが様々な方法でメールボックスの一覧に表示される。

【００１７】

親出願に開示されているいくつかの機能はあらゆるＥメールクライアントソフトウェアに適用可能であり、それらを使用してＥメールへのファイルの添付とＥメールに添付されたファイルの使用プロセスを改善することができる。

【００１８】

（発明の要約）

従って、異なる種類のメッセージと添付文書を異なる種類のアイコンと共に表示する受信ボックスリストを含む電子メールプログラムを提供することが本発明の目的である。

【００１９】

以下に詳述するこの目的に従い、本発明による電子メール・クライアント・ソフトウェアは、メッセージを各メッセージ用のアイコンと共に一覧表示するメールボックス・ディスプレイヤを備えており、それらの各アイコンは、各メッセージのＭＩＭＥタイプと関連付けられている。添付ファイルを含むメールは、そのＥメールに添付された当該ファイルのタイプを示すアイコンと共に受信ボックスに一覧表示される。メールボックス・ディスプレイヤは、ＭＩＭＥタイプを解釈し、ＯＳ内のアイコンレジストリ又はＥメール・クライアント・ソフトウェアが維持管理するアイコン・ディレクトリから適切なアイコンを選択する。例えば、ＡＤＯＢＥ　ＡＣＲＯＢＡＴファイルが添付されたＥメールを受信すると、メールボックの一覧の当該メールアイテムのそばにＡＤＯＢＥ　ＡＣＲＯＢＡＴアイコンが表示される。更に、親出願で説明されているように特別なオーサリング／読み取りコンポーネントによってメッセージが作成されている場合には、そのオーサリング／読み取りコンポーネントに関連付けられたアイコンがメールアイテムを表示する行の一部としてメールボックスの一覧に表示される。

【００２０】

本発明の電子メールソフトウェアは、親出願のＥメールソフトウェアを参照して例を挙げて説明するが、このＥメールソフトウェアには、メインＥメールコンポーネントと、このＥメールコンポーネントとアプリケーションプログラムインターフェイス（ＡＰＩ）を介して双方向に通信するいくつかのインストール可能コンポーネントが含まれている。そして、インストール可能コンポーネントには、オーサリング／読み取りコンポーネント及び

メールボックスディスプレイヤコンポーネントが含まれている。又、好適な実施例によれば、アイコンのデータベースを維持管理するためのコンポーネントも含まれる。

【００２１】

メールボックス・ディスプレイヤ・コンポーネント機能は、メールボックスを開いたりメールのリストをスクロールしたりする際に、ユーザーによって起動される。メールボックス・ディスプレイヤ・コンポートネントには、最新技術のメールボックス・ディスプレイヤのすべての機能を含むと共に、メッセージのＭＩＭＥタイプに基づいてアイコンディレクトリをチェックしアイコンを当該メッセージに関する情報と共に表示する機能を含むことが好ましい。ＬＩＮＧＯによる実施例では、各メッセージごとに、メッセージのＭＩＭＥタイプとサブタイプに基づいた追加のＴＹＰＥフィールドを有するデータ構造が生成され、この内部ＴＹＰＥフィールドを使用してＭＩＭＥタイプをアイコンに関連付けている。別の実施例では、アイコン画像と関連付けるために、メッセージの「ｃｏｎｔｅｎｔ－ｔｙｐｅ」（ＭＩＭＥタイプ）ヘッダの内容を直接使用している。そして、アイコンディレクトリに適切なアイコンが存在しない場合には、メールボックスディスプレイヤは、ＭＩＭＥメッセージのサブパートに格納されているアイコン画像データを使用する（このデータが提供されている場合）。そうでなければ、アイコンを使わないか、或いは、一般アイコンを使用する。好適な実施例によれば、タイプアップデータコンポーネントによってタイプテーブル（ＴＹＰＥＴＡＢＬＥ）が維持管理される。このタイプテーブルには、メッセージタイプ及びサブタイプのリストが、メールボックスディスプレイヤが使用する拡大縮小可能なアイコンのファイル名と共に含まれている。本発明には、メールボックスの内容の表示用に選択されたフォントサイズと整合するよう、アイコンのサイズを変更できる拡大縮小可能なアイコンが好ましい。

【００２２】

タイプアップデータコンポーネントの複数の実施例が提供されている。第１実施例によれば、アイコンは、ユーザーが手作業でインストール／削除する。第２実施例によれば、アイコンは、モジュラーオーサリング／読み取りコンポーネントがインストール／削除される際に自動的にインストール／削除される。第３実施例によれば、メールボックスディスプレイヤが新しいメッセージタイプを検出するごとに新しいアイコンが自動的に追加され、この新しいアイコンは、オペレーティングシステムのレジストリ又はメッセージに組み込まれているアイコン画像データから取得される。そして、第４実施例によれば、タイプアップデータが新しいアイコン情報について自動的にネットワークサーバーに問い合わせ、必要に応じてアイコン画像データをダウンロードする。

【００２３】

本発明の更なる目的と利点は、添付の図面と関連して提示する以下の詳細な説明を参照することにより、当業者に明らかになるであろう。

【００２４】

（好適な実施例の詳細な説明）

図２を参照すれば、本発明による電子メールクライアントソフトウェアは、メッセージを各メッセージ用のアイコンと共に一覧表示するメールボックスディスプレイヤを備えており、これらの各アイコンは各メッセージのＭＩＭＥタイプと関連付けられている。ファイルが添付されているメールは、そのＥメールに添付されたファイルのタイプを示すアイコンと共にこの受信ボックスに一覧表示される。例えば、図２に示すように、「議事録（Ｍｉｎｕｔｅｓ　ｏｆ　Ｍｅｅｔｉｎｇ）」という件名のメールメッセージは、このＥメールメッセージにＷＯＲＤファイルが添付されていることを示すＭＩＣＲＯＳＯＦＴ　ＷＯＲＤアイコンと共に一覧表示されている。そして、「新しいラジオ広告」という件名のメッセージは、このＥメールにオーディオファイルが添付されていることを示すＱＵＩＣＫＴＩＭＥ　ＷＡＶアイコンと共に一覧表示されており、「新しい広告パンフレット」という件名のメッセージには、ＡＣＲＯＢＡＴアイコンが示しているように、ＡＤＯＢＥ　ＡＣＲＯＢＡＴファイルが添付されている。同様に「売上予測」メッセージは、ＥＸＣＥＬ表計算アイコンと共に表示され、「顧客データベース」メッセージは、ＦＩＬＥＭＡＫＥ

Ｒ　ＰＲＯデータベースアイコンと共に表示され、「年末会計」メッセージは、ＱＵＩＣＫＥＮアイコンと共に表示されている。本発明によれば、メールボックスディスプレイヤは、メッセージのＭＩＭＥタイプ及び／又は添付ファイル（存在する場合）のＭＩＭＥタイプ又は文書タイプを解釈し、ＯＳ内のアイコンレジストリ又はＥメールクライアントソフトウェアが維持管理するアイコンディレクトリから適切なアイコンを選択する。

【００２５】

本発明の電子メールソフトウェアは、先に包含された親出願のＥメールソフトウェアを参照して例を挙げて説明するが、このＥメールソフトウェアには、メインＥメールコンポーネントと、このＥメールコンポーネントとアプリケーションプログラミングインターフェイス（ＡＰＩ）を介して双方向に通信するいくつかのインストール可能コンポーネントが含まれている。そして、インストール可能コンポーネントには、オーサリング／読み取りコンポーネントと少なくとも１つのメールボックスディスプレイヤコンポーネントが含まれる。図２ａは、先程包含された親出願のＥメールソフトウェアの実施例を示しており、メールメッセージに関連付けられたオーサリング／読み取りコンポーネントを示すアイコンが表示されている。この実施例は、従来技術のＥメールクライアントからＥメールを受信すると共に、それらに対してＥメールを送信する能力を有している。図２ａには、２つのアイコン（即ち、ペーパークリップ及びＡＣＲＯＢＡＴアイコン並びにペーパークリップ及びＰＨＯＴＯＳＨＯＰアイコン）を有する２つのメールアイテムが表示されたメールボックスリストが示されている。これらのデュアルアイコン表示は、当該メールが特別なオーサリング／読み取りコンポーネントなしに作成され、その他のプログラム（この場合には、ＡＣＲＯＢＡＴ及びＰＨＯＴＯＳＨＯＰ）によって作成された添付ファイルを備えていることを示している。

【００２６】

前述のように、本発明の電子メールソフトウェアは、先程包含された親出願のＥメールソフトウェアを参照して例を挙げて説明するが、このＥメールソフトウェアには、メインＥメールコンポーネントといくつかのインストール可能コンポーネントが含まれている。又、好適な実施例によれば、アイコンのデータベースを維持管理するためのコンポーネントも含まれており、図３に、これらのコンポーネントの間の関係が示されている。

【００２７】

この図３に示すように、ＴＹＰＥＴＡＢＬＥと呼ばれるデータ構造１０がタイプアップデータ（ＴＹＰＥ　ＵＰＤＡＴＥＲ）と呼ばれるコンポーネント１２によって作成され、維持管理される。本発明によるタイプアップデータの実用的な例が付属書Ｂに示されているが、これについては後程詳細に説明する。ＴＹＰＥＴＡＢＬＥの主な目的は、メールボックスディスプレイヤコンポーネント１４によって読み取られることである。タイプアップデータ１２とメールボックスディスプレイヤ１４は、関数の呼び出しと共有データ構造ＴＹＰＥＴＡＢＬＥを使用して互いに通信する。例えば、メールボックスディスプレイヤは、タイプ・アップデータ・コンポーネント内の「ｉｎｉｔｉａｌｉｚｅ＿ＴＹＰＥＴＡＢＬＥ」関数（図６ａの３０６）に対する呼び出しを含んでいる。メールボックスディスプレイヤは、親出願及び先程包含された出願第ＢＡＫ－００７号に説明されているＡＰＩを使用して電子メールクライアント・ソフトウェアと通信する。図３に示すように、メールボックスディスプレイヤコンポーネント１４と電子メールクライアントソフトウェア１６もユーザーのメールボックス内の電子メールメッセージデータに対して双方向にアクセスしている。好適な実施例によれば、親出願に説明されている種類のインストール可能なアプリケーションコンポーネントによってメッセージが作成されている場合には、電子メールメッセージは、そのＭＩＭＥタイプ及びサブタイプヘッダフィールドから導出された情報を含む任意選択のタイプフィールドと共に保存される。

【００２８】

メールボックスディスプレイヤコンポーネント機能は、メールボックスを開いたりメールのリストをスクロールした際に、ユーザーによって自動的に起動される。当業者であれば、大部分の電子メールクライアントソフトウェアは、受信ボックス、送信ボックス、読み

取り済みメール、送信済みメールなどのいくつかの異なるメールボックスを備えていることを理解するであろう。メールボックスディスプレイヤコンポーネント１４は、最新技術のメールボックスディスプレイヤのすべての機能を含むと共に、ＴＹＰＥＴＡＢＬＥ（及び、後述するように、ＭＩＭＥメッセージの本文）をチェックして適切なアイコンを探し出してメールボックス表示の当該メッセージの件名のそばに表示する機能を含むことが好ましい。好適な実施例によれば、アイコンは小さな画像ファイル（例：ＥＰＳファイル又はＧＩＦファイル）で保存され、ＴＹＰＥＴＡＢＬＥデータ構造によってポイントされている。又、メールボックス表示において様々なサイズのフォントを使用できるよう、アイコングラフィックのサイズを拡大縮小する手段も提供される。これには、３つの方法が使用可能であり、まず、標準的な補間アルゴリズムを使用して画像を拡大縮小することができる。２番目として、解像度が異なる複数のアイコン画像のコピーを保存しておき、限られた数のフォントポイントサイズに整合するように取得してもよい。更には、３番目として（現状ではこれが好ましい）、以上の２つの方法を組み合わせて使用し、各アイコンごとに少なくとも１つの画像を保存し、フォントのポイントサイズに最も密接に整合するアイコンを選択した後に、必要に応じて拡大縮小し、フォントポイントサイズに更に整合させるのである。

【００２９】

表１は、ＴＹＰＥＴＡＢＬＥデータを永久ストレージ（例：ハードディスク）に保存する方法を示している。

【００３０】

【表１】

ＭＩＭＥタイプ	アイコンのファイル名	メッセージハンドラのファイル名
text/plain	c:¥kidcod¥text.gif	c:¥kidcode¥txt.dxr
x-application/rebus	c:¥kidcode¥rebus.gif	c:¥kidcode¥rebus.dxr
x-application/grid	c:¥kidcode¥grid.gif	c:¥kidcode¥grid.dxr
x-application/graph	c:¥kidcode¥graph.gif	c:¥kidcode¥grph.dxr
...	...	...
multipart/mixed	c:¥kidcode¥paperclip.gif	

【００３１】

この表１には、少なくとも５つのＭＩＭＥタイプが示されている。最初の４つは、オーサリング／読み取り用のインストール可能コンポーネントを利用するＭＩＭＥタイプである。インストール可能コンポーネントは、「．ｄｘｒ」ファイル拡張子によって示されている。表１に示されているｍｕｌｔｉｐａｒｔ／ｍｉｘｅｄ（５番目）ＭＩＭＥタイプは、インストール可能コンポーネントではなく、外部アプリケーションによって作成された添付ファイルを示している。図２ａを参照して述べたように、インストール可能コンポーネントで作成されたメールメッセージとこの添付ファイルを一般的なペーパークリップアイコンを使用して区別している。

【００３２】

表２は、ＲＡＭにロードされる際のＴＹＰＥＴＡＢＬＥデータ構造を示している。

【００３３】

【表２】

ＭＩＭＥタイプ	ポインタ	アイコンのファイル名	メッセージハンドラのファイル名
text/plain	20	c:¥kidcod¥text.gif	c:¥kidcode¥txt.dxr
x-application/rebus	21	c:¥kidcode¥rebus.gif	c:¥kidcode¥rebus.dxr
x-application/grid	22	c:¥kidcode¥grid.gif	c:¥kidcode¥grid.dxr
x-application/graph	23	c:¥kidcode¥graph.gif	c:¥kidcode¥grph.dxr
...		...	...
multipart/mixed	19	c:¥kidcode¥paperclip.gif	

【００３４】
ＴＹＰＥＴＡＢＬＥデータ構造をＲＡＭにロードしたものはＳＧ＿ＴＹＰＥＴＡＢＬＥと呼ばれ、図２に示す構造を有している。この構造には、アイコンに対するポインタが含まれており、模範的な実施例におけるこのポインタは、ＬＩＮＧＯのキャストメンバである。

【００３５】
前述のように、模範的な実施例によれば、Ｅメールメッセージが添付ファイルを有している場合には、一般的な添付ファイルアイコンが表示されると同時に、当該添付ファイルに固有のアイコンも表示される。この２番目のアイコンは、オンザフライで作成され（例えば、図６ｂに示されているように）、ＳＧ＿ＡＴＴＡＣＨ＿ＴＹＰＥＴＡＢＬＥと呼ばれるＲＡＭ内のデータ構造によってポイントされる。このＳＧ＿ＡＴＴＡＣＨ＿ＴＹＰＥＴＡＢＬＥの基本構造が表３に示されている。

【００３６】
【表３】

ファイル拡張子	アイコンポインタ	プログラムのファイル名
.doc	30	c:¥programs¥winword.exe
.pdf	31	c:¥programs¥acrobat.exe
.html	32	c:¥programs¥netscape.exe
.htm	32	c:¥programs¥netscape.exe
.xml	32	c:¥programs¥netscape.exe
...	...	...

【００３７】
この表３に示すように、ファイル拡張子がアイコンポインタと当該添付ファイルの読み取りに使用するプログラムへのパス名に関連付けられている。図６ｂを参照して以下に詳述するように、このＳＧ＿ＡＴＴＣＨ＿ＴＹＰＥＴＡＢＬＥは、システムレジストリのアイコンを使用してオンザフライで作成される。

【００３８】
メールボックスディスプレイヤコンポーネントの好適な実施例が付属書Ａに詳細に定義されているが、この付属書Ａは、先程包含された親出願の付属書Ｂと類似している。本明細

書の付属書Ａに示すコードリストは、メインメールボックス表示関数が始まる第２８７行以降において親出願のコードリストと異なっており、このメインメールボックス表示関数は、図４のフローチャートにも示されている。ここでこの図４と付属書Ａを参照する。この関数は、付属書Ａでは第２８７行及び図４のフローチャートの開始点１０１から始まっている。まず、メールボックスの内容を表示する前に、付属書Ａの第２９３～３００行及び図４の１０３、１０５に示すように、ＴＹＰＥＴＡＢＬＥへのポインタと関連システム機能がセットアップされる。そして、様々なメールボックスフィールド（例：メッセージ番号、件名、日付、メッセージ読み取りインジケータ）が付属書Ａの第３０２～３０７行及び図４の１０７においてクリアされる。付属書Ａの第３０９～３１１行及び図４の１０７において、アイコングラフィックスの表示に使用される「スプライトチャネル」（ＭＡＣＲＯＭＥＤＩＡグラフィックホルダ）がクリアされる。次に、付属書Ａの第３１３行以降及び図４の１０９において、メッセージリストが表示される。メールボックス表示の各行を構成する要素（例：メッセージの件名、日付、送信者の名前、並びにグラフィック要素）は「プロパティ（属性）」と呼ばれるが、付属書Ａの第３２０～３２６行及び図４の１１１、１１３において、これらのプロパティがメッセージから読み取られる。図４の１１１及び付属書Ａの第３２０～３２３行に示されているように、メールボックス、ＭＩＭＥタイプ、及びステータスを除き、すべてのプロパティは、読み取られると自動的に表示される。一方、メールボックス、ＭＩＭＥタイプ、及びステータスの各プロパティは、図４の１１３及び付属書Ａの第３２４～３２６行において読み取っている。好適な実施例によれば、ＭＩＭＥタイプアイコンは、メッセージのステータスの表示にも使用されており、メッセージが既読の場合には、アイコンはグレースケールで表示されるが、メッセージが未読の場合は、アイコンは色付きで表示される。アイコンを見つけ出し、そのアイコンをグレースケール又は色付きに設定する段階が付属書Ａの第３２８～３５５行に示されている。付属書Ａの第３３２～３３４行及び図４の１１５でスプライトチャネルカウンタが設定され、ＴＹＰＥＴＡＢＬＥデータ構造が付属書Ａの第３４０～３４３行及び図４の１１７において解析される。このメッセージのＭＩＭＥタイプを示すアイコンがＴＹＰＥＴＡＢＬＥに含まれていない場合は、付属書Ａの第３４４～３４６行及び図４の１１９において既定のアイコンが選択される。当該メッセージのステータスがメッセージ（その添付ファイル）が既読であることを示している場合には、付属書Ａの第３４８～３４９行及び図４の１２１においてグレースケールバージョンのアイコンが設定される。そして、メールボックスの内容リストをスクロールする際に迅速にアクセスできるよう、選択されたアイコンがアイコンリストに追加されるが、これは付属書Ａの第３５１～３５２行及び図４の１２３に示されている。付属書Ａの第３５４～３７４行及び図４の１２５のメインメールボックス表示関数の残りの部分は、適切な画面位置へのテキストとアイコンの配置に関連するものである。尚、第３１５～３７４行に示されているコードは、図４の１２７の判定が示しているように、メールボックス内のメッセージの数だけ繰り返される。そして、一覧表示するメッセージがなくなると、メインメールボックス表示関数は、付属書Ａの第３７６行及び図４の１２９に示すように終了する。

【００３９】

ここでは、これらのコード又はフローチャートには示されていないが、前述のように、アイコンのディレクトリに適切なアイコンが存在しない場合には、メールボックスディスプレイヤは、ＭＩＭＥメッセージのサブパートに格納されているアイコン画像データを使用する（このデータが提供されている場合）。当業者であれば、前述の規格によって定義されているＭＩＭＥファイル内の場所からアイコン画像データを読み取ることにより、この機能を簡単に実行できることを理解するであろう。これを実装するコードは、付属書Ａの第３２５行又は付属書Ａの第３４５行に挿入可能である。

【００４０】

前述のように、付属書Ａの残りの部分は、親出願の付属書Ｂのメールボックスコンポーネントと実質的に同一であり、それらの説明は先程包含された親出願に十分に示されている。当業者であれば、これら付属書のコードリストがＭＡＣＲＯＭＥＤＩＡ　ＤＩＲＥＣＴ

ＯＲ開発スイートに固有のものであり、別の開発環境を使用しても同一の機能を実現することができることを理解するであろう。図５は、このメールボックス表示関数の機能を、別のプログラミング言語に適用可能な更に一般化した形態で示している。

【００４１】

この図５を参照すれば、メールボックスの表示は、２０１において始まり、メッセージのリストが読み取られる。次に２０２において、ＴＹＰＥＴＡＢＬＥが既に初期化されているかどうかをチェックし、判定する。そして、初期化されていなければ、図７を参照して以下に詳述するように、２０４においてＴＹＰＥＴＡＢＬＥが初期化される。２０３において、メールボックス表示は次のメッセージを取得する。そして、２０３で取得されたこのメッセージに関し、メールボックスは、２０５においてそのプロパティ、２０７においてそのプロパティ値を取得する。そして、２０９において、そのプロパティが表示用アイコンを使用すると判定された場合には、２１１において「アイコン取得」ルーチンが呼び出される（このルーチンは図６に示されている）。このアプリケーションに関係するメッセージプロパティは、（ａ）メッセージのＭＩＭＥタイプ、及び（ｂ）メッセージにファイルが添付されているかどうか、の２つである。図５のフローチャート（２０９）において、本システムは、ＬＩＮＧＯのシンボルとして実装されている当該メッセージのプロパティ（例：＃ｔｙｐｅ、＃ｄａｔｅ、＃ｍａｉｌｂｏｘ、＃ｓｔａｔｕｓ）がアイコンによって表されるかどうかをルックアップする。この実装では、各プロパティが個別にコーディングされているが、図５のフローチャートが説明する更に一般的な実装では、ターゲットシンボル（例：＃ｔｙｐｅ）がリストデータ構造の要素であるかどうかを図５の手順２０９においてチェックすることにより（例：Ｐｒｏｐｅｒｔｉｅｓ＿ｗｉｔｈ＿Ｉｃｏｎｓ＝（＃ｔｙｐｅ，＃ｈａｓ＿ａｔｔａｃｈｍｅｎｔ）実現することができる。そして、当該アイコンが２１３で表示される。一方、２０９で、プロパティがアイコンを使用しないと判定された場合には、２１５においてプロパティの値が表示される。そして、２１７において、このメッセージに更なるプロパティが存在するかどうかが判定され、存在する場合には、プログラムは２０５に戻るが、存在しない場合には、２１９において一覧表示すべき更なるメッセージが存在するかどうかが判定される。そして、存在する場合には、プログラムは２０３に戻り、存在しなければ、プログラムは２２１で終了する。

【００４２】

図６は、付属書Ａの第３２８～３５０行に含まれている機能を一般化したものを示している。これは、図５の２１１で呼び出される「アイコン取得」ルーチンを一般化したものである。このルーチンは、３０１において、呼び出し元のプログラムからプロパティ名とプロパティ値を取得して始まる。そして、３０３において、プロパティが「ＭＩＭＥタイプ」である、即ち、ＬＩＮＧＯ実装の「＃ｔｙｐｅ」に一致していると判定されると、３０５においてＭＩＭＥタイプのアイコンが取得され、３０７においてそのアイコンへのポインタが呼び出し元のプログラムに返される。このＭＩＭＥタイプのアイコン取得に必要な実際の段階は図６ａに示されている。３０９において、プロパティが「添付ファイルが存在する（ｈａｓ　ａｔｔａｃｈｍｅｎｔ）」であると判定されると、「添付ファイルが存在する」のアイコンが３１１において取得され、３０７において、そのアイコンへのポインタが呼び出し元のプログラムに返される。Ｍｉｃｒｏｓｏｆｔ　Ｗｉｎｄｏｗｓプラットフォーム用のファイル添付アイコンの取得手順が図６ｂで説明されている。この代わりに、ファイル拡張子からＭＩＭＥタイプ及びＭＩＭＥタイプアイコンへのマッピングに使用するファイルタイプ用のフィールドをＴＹＰＥＴＡＢＬＥデータ構造に含めてもよい。この方法でＴＹＰＥＤＡＴＡを変更した場合には、メッセージタイプ及びファイルタイプのアイコンルックアップの両方でＴＹＰＥＴＡＢＬＥを使用することになる。これに関連し、従来未知のメッセージタイプ又はファイルタイプと検出した際にそれらのファイルタイプ／ＭＩＭＥタイプの関連付けをインストールするためのＴＹＰＥ＿ＵＰＤＡＴＥＲの変更も必要になろう。又、ファイルタイプ拡張子は、ＭＩＭＥタイプほどに豊富ではないため、同一のファイルタイプ拡張子が多数の異なるＭＩＭＥタイプにマッピングされることになる。例えば、ソフトウェア製造者が彼らのソフトウェアのバージョンごとに異なる

ＭＩＭＥサブタイプを割り当てた場合、その同一ソフトウェアの異なるバージョンにおいて、このような状況が生じることになる。

【００４３】

３１３において、プロパティが「既読（ｍｅｓｓａｇｅ　ｒｅａｄ）」であると判定された場合は、「既読」用のアイコンが３１５で取得され、３０７において、そのアイコンへのポインタが呼び出し元のプログラムに返される。そして、３１７において、プロパティが「優先順位（ｐｒｉｏｒｉｔｙ）」であると判定されると、「優先順位」用のアイコンが３１９で取得され、３０７においてそのアイコンへのポインタが呼び出し元のプログラムに返される。以上の内容から、当業者であれば、様々なプロパティに対して多数の異なるアイコンを表示することができることを理解するであろう。

【００４４】

図６ａは、付属書Ａの第３２８～３４６行に含まれている実装を一般化したものを示している。これは、図６の３０５で呼び出される「ＭＩＭＥタイプ用アイコン取得（ｇｅｔ　ｉｃｏｎ　ｆｏｒ　ｍｉｍｅｔｙｐｅ）」関数である。このルーチンは、呼び出し元のプログラムから「ＭＩＭＥタイプ」を与えられ、３０２において始まる。まず３０４において、ＴＹＰＥＴＡＢＬＥが既に初期化されているかどうかを判定する。そして、初期化されていない場合は、３０６において初期化する。この初期化ルーチンは、タイプアップデータコンポーネントに存在するか、或いは、タイプアップデータコンポーネントに存在する関数を呼び出す形態であってもよい。例えば、ＬＩＮＧＯの実装においては、タイプアップデータコンポーネントの一部である関数Ｒｅａｄ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅ（図８）を使用してＴＹＰＥＴＡＢＬＥデータ構造を初期化している。そして、この初期化の後に、或いは、３０６においてＴＹＰＥＴＡＢＬＥが既に初期化済みであると判定された場合には、３０８において、ＭＩＭＥタイプを使用してＴＹＰＥＴＡＢＬＥからアイコンポインタを取得する。付属書Ａの第３２８～３４２行を参照されたい。次に３１０において、そのアイコンポインタがＮＵＬＬ（空）であるかどうかを判定する。そして、ＮＵＬＬである場合には、３１２において、「メッセージタイプインストール（ｉｎｓｔａｌｌ　ｍｅｓｓａｇｅ　ｔｙｐｅ）」ルーチンが呼び出される。タイプアップデータコンポーネントに含まれているこの「メッセージタイプインストール」ルーチンについては、ＭＩＭＥタイプ用の新しいメッセージハンドラとアイコンのインストールを示す図１１を参照して後程詳細に説明する。一方、アイコンポインタがＮＵＬＬでなければ、３１４において、アイコンポインタが「アイコンがインストールされていない（ｉｃｏｎ　ｎｏｔ　ｉｎｓｔａｌｌｅｄ）」をポイントしているかどうかを判定する。そして、これをポイントしている場合には、３１６において、アイコンポインタを既定のポインタに設定するが、いずれの場合にも、３１８において、ＮＵＬＬではないポインタが呼び出し元のプログラムに返される。

【００４５】

図６ｂは、Ｅメールメッセージの添付ファイル用のアイコンを取得する模範的な手順を示している。３２０で始まり、まず添付ファイルのファイル拡張子を読み取る（Ｗｉｎｄｏｗｓプラットフォームの場合には、ファイル名の「．」に続く３文字によって「ファイルタイプ」が決定されるが、Ｍａｃｉｎｔｏｓｈプラットフォームなどのその他のプラットフォームでは、ファイルタイプと「クリエータコード」が、ファイルの「リソースフォーク」に一覧表示されている。従って、これらのプラットフォームの場合には、まず、ファイルのリソースフォークからファイルタイプ（及びクリエータコード）を読み取ることに留意されたい）。このファイルタイプ（又はファイルタイプとクリエータコード）の判定が完了すると、このルーチンは、３２２においてＳＧ＿ＴＹＰＥＴＡＢＬＥ内で適切なアイコンを見つけ出そうと試みる。そして、３２４において適切なアイコンが見つけられなかったと判定された場合には、３２６において、ＳＧ＿ＡＴＴＡＣＨ＿ＴＹＰＥＴＡＢＬＥ内で適切なアイコンを見つけ出そうと試みる。そして、３２８において、再度適切なアイコンが見つけられなかったと判定されると、ルーチンは、３３０及び３３２において、システムレジストリ内で適切なアイコンを見つけ出そうと試みる（その他のオペレーティ

ングシステムでは、アイコンリソースが別の場所に保存されている場合がある。例えば、Ｍａｃｉｎｔｏｓｈ　ＯＳの場合には、アイコンリソースは、非表示の「デスクトップ」ファイル内に保存されていることに留意されたい）。３３４において適切なアイコンが検出された場合には、３３６においてそのアイコンへのポインタが設定され、そのポインタが３３８においてＳＧ＿ＡＴＴＡＣＨ＿ＴＹＰＥＴＡＢＬＥに書き込まれる。そして、３４０において、そのアイコンポインタがメールボックスディスプレイヤに返される。図６の３１１と図５の２１１を参照されたい。一方、３４２において適切なアイコンが検出されない場合には、アイコンポインタは提供されない。

【００４６】

次に付属書Ｂを参照すれば、第１～２６行には、タイプアップデータの概要と紹介情報が提供されている。このタイプアップデータには、１１個の関数が含まれている。この中の３つは、メールボックスディスプレイヤによって呼び出されるパブリック関数であり、これらには、Ｉｎｉｔｉａｌｉｚｅ＿ＴｙｐｅＴａｂｌｅ、Ｉｎｓｔａｌｌ＿Ｔｙｐｅ、及びＵｎｉｎｓｔａｌｌ＿Ｔｙｐｅが含まれる。そして、残りの８つの関数は、タイプアップデータ内部で使用されるプライベート関数であり、これらには、Ｗｒｉｔｅ＿Ｔｙｐｅｔａｂｌｅ、Ｒｅａｄ＿Ｔｙｐｅｔａｂｌｅ＿Ｆｉｌｅ、Ｒｅａｄ＿Ｉｃｏｎ＿Ｆｉｌｅｓ＿Ｔｏ＿ＲＡＭ、ｒｅａｄ＿ｉｃｏｎＦｉｌｅ、ｄｅｌｅｔｅ＿ｍｉｍｅｔｙｐｅ、ｉｎｓｅｒｔ＿ｍｉｍｅｔｙｐｅ、ｄｅｌｅｔｅ＿ｆｉｌｅｔｙｐｅ、ｉｎｓｅｒｔ＿ｆｉｌｅｔｙｐｅが含まれる。まず、Ｉｎｉｔｉａｌｉｚｅ＿ＴｙｐｅＴａｂｌｅ関数は、図７及び付属書Ｂの第２９～６０行に示されている。この関数は図７の３６０において始まり、まず３６２においてＳＧ＿ＴＹＰＥＴＡＢＬＥをＮＩＬ（空）に設定する（付属書Ｂの第３９行）。次に３６４において、ＳＧ＿ＡＴＴＡＣＨ＿ＴＹＰＥＴＡＢＬＥをＮＩＬに設定する（付属書Ｂの第４３行）。そして、３６６においてディスク上に保存されているＴＹＰＥＴＡＢＬＥファイルがＳＧ＿ＴＹＰＥＴＡＢＬＥに読み込まれる（付属書Ｂの第４４行）。このｒｅａｄ＿ｔｙｐｅｔａｂｌｅ＿ｆｉｌｅ内部関数は、図８及び付属書Ｂの第２０７～２５２行に示されている。３６８（付属書Ｂの第４６行）においてエラーチェックが実行され、データがＲＡＭにロードされたかどうかを判定する。そして、データが読み込まれていない場合には、３７０（付属書Ｂの第４７行）においてユーザーに警告し、３７２（付属書Ｂの第４８行）において失敗（ｆａｉｌｕｒｅ）を返す。一方、データが読み込まれている場合には、３７４（付属書Ｂの第５１行）においてアイコンファイルがＲＡＭに読み込まれる。そして、３７６（付属書Ｂの第４７行）においてエラーチェックが実行され、データの読み込みに成功していれば、３７８（付属書Ｂの第６０行）においてこの関数は終了する。

【００４７】

次に図８及び付属書Ｂの第２０７～２５２行を参照すれば、ｒｅａｄ＿ｔｙｐｅｔａｂｌｅ＿ｆｉｌｅ関数は３８０（付属書Ｂの第２１８行）において始まる。まず３８２（付属書Ｂの第２２２行）において、ＳＧ＿ＴＹＰＥＴＡＢＬＥをＮＩＬに設定する。そして、３８４（付属書Ｂの第２２４より２２７行）でｔｙｐｅｔａｂｌｅ．ｔｘｔファイルを開き、３８６（付属書Ｂの第２２９行）においてファイルのオープンエラーのチェックを行う。そして、ファイルのオープンエラーが検出された場合には、３８８（付属書Ｂの第２３０行）においてエラー警告を生成し、３９０（付属書Ｂの第２３１行）でファイルを閉じ、３９２（付属書Ｂの第２３２行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥを返す。一方、エラーなしにファイルがオープンされた場合には、３９１において最初の行を読み込むが、ファイルの末尾に到達した際に最終的に３９４で終了するループ（第３９４～４０８行）がこの３９１から始まっている。このループが終了すると、３９０（付属書Ｂの第２３１行）においてファイルが閉じられ、３９２（付属書Ｂの第２３２行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥが返される。尚、付属書Ｂの第２３５～２３９行に示すように、ＬＩＮＧＯの場合には行単位の読み取りができないため、このＬＩＮＧＯによる実装では、ファイル全体を１つの文字列に読み込んで行単位の読み取りをシミュレートしている。１つの行を読み取ると、その行の最初のワード（データ）がＭＩＭＥタイプである（付属書Ｂの

第２４３行）。このＭＩＭＥタイプを３９６（付属書Ｂの第２４４行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥに挿入する。そして、３９８において、ＭＩＭＥタイプが定義されていないことが判明した場合には、４００でユーザーに警告し、３９１において次の行が読み込まれる。一方、ＭＩＭＥタイプが定義されている場合には、４０２（付属書Ｂの第２４５行）において、その行の次のワード（ファイルタイプ）が読み込まれ、ＳＧ＿ＴＹＰＥＴＡＢＬＥに挿入される。続いて、４０４（付属書Ｂの第２４６行）において、行内の次のワード（ｉｃｏｎＦｉｌｅＮａｍｅ）が読み込まれてＳＧ＿ＴＹＰＥＴＡＢＬＥに挿入され、４０６（付属書Ｂの第２４７行）において行内の次のワード（ｍｓｇＨａｎｄｌｅｒ）が読み込まれてＳＧ＿ＴＹＰＥＴＡＢＬＥに挿入される。４０８において検出されたすべてのエラーが４００において報告されるが、前述のように、このプロセスは、ｔｙｐｅｔａｂｌｅ．ｔｘｔファイルのすべての読み取りが完了するまで継続される。

【００４８】

関数Ｗｒｉｔｅ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅが図９及び付属書Ｂの第１５９～２０６行に説明されている。この関数は、図１１を参照して後述するＩｎｓｔａｌｌ＿Ｔｙｐｅ関数によって新しいＭＩＭＥタイプとアイコンがＳＧ＿ＴＹＰＥＴＡＢＬＥに追加された後に、そのＳＧ＿ＴＹＰＥＴＡＢＬＥの内容をｔｙｐｅｔａｂｌｅ．ｔｘｔに書き戻す。このＷｒｉｔｅ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅ関数は４１０（付属書Ｂの第１６３行）において始まり、まず４１２（付属書Ｂの第１６８行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥ構造がＮＩＬであるかどうかをチェックする。そして、ＮＩＬの場合には、４１４（付属書Ｂの第１６９行）においてエラーメッセージを生成し、４１６（付属書Ｂの第１７０行）において「失敗」メッセージを返す。一方、ＳＧ＿ＴＹＰＥＴＡＢＬＥ構造がＮＩＬでなければ、４１８（付属書Ｂの第１７３行）においてｔｙｐｅｔａｂｌｅ．ｔｘｔファイルのバックアップコピーを作成し、４２０（付属書Ｂの第１７６～１８０行）において新しい空のファイルが作成される。そして、４２２（付属書Ｂの第１８１行）においてエラーチェックが実行され、新しいファイルの作成にエラーが検出された場合には、４２４（付属書Ｂの第１８２行）においてエラーメッセージが返され、４２６（付属書Ｂの第１８３行）でバックアップファイルが回復された後に、４２８（付属書Ｂの第１８４行）において戻りコードが「失敗」に設定される。そして、４３０（付属書Ｂの第２０３行）でｔｙｐｅｔａｂｌｅ．ｔｘｔファイルが閉じされ、４３２（付属書Ｂの第２０４行）においてバックアップファイルを削除した後に、４３４（付属書Ｂの第２０５～２０６行）において戻りコードを返す。一方、新しいファイルの作成にエラーがなければ、第１８７～１８８行においてそのファイルを書き込みアクセス用に開き、第１９１行において開始行カウンタを設定する。そして、４３６（付属書Ｂの第１９２行）でＳＧ＿ＴＹＰＥＴＡＢＬＥ内の最初の（次の）エントリを読み込み、４３８（付属書Ｂの第１９３行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥの末尾に到達したと判定された場合には、４４０（付属書Ｂの第２０１行）において戻りコードを「成功（ｓｕｃｃｅｓｓ）」に設定し、４３０でファイルを閉じ、４３２でバックアップを削除した後に、４３４で戻りコードを返す。

【００４９】

ＳＧ＿ＴＹＰＥＴＡＢＥの末尾に到達するまでは、４４２において書き込みデータを設定され、４４６（付属書Ｂの第１９４行）においてＭＩＭＥタイプが文字列に書き込まれる。更に、４４６、４４８、及び４５０（付属書Ｂの第１９５～１９７行）において、ｆｉｌｅｔｙｐｅ、ｉｃｏｎｆｉｌｅｎａｍｅ、及びｍｅｓｓａｇｅｈａｎｄｌｅｒｆｉｌｅｎａｍｅが、それぞれ文字列に追加される。そして、４５２（付属書Ｂの第１９８行）において、その文字列を新しいｔｙｐｅｔａｂｌｅ．ｔｘｔファイルに書き込み、４５４（付属書Ｂの第１９９～２００行）において行区切り文字を書き込んだ後に、この関数は４３６にループして戻るが、これらの動作は、ＳＧ＿ＴＹＰＥＴＡＢＬＥ内のすべてのエントリが読み込まれ、新しいｔｙｐｅｔａｂｌｅ．ｔｘｔファイルに書き込まれるまで継続する。

【００５０】

図１０は、付属書Ｂの第２５５～２９０行に示されている関数Ｒｅａｄ＿Ｉｃｏｎ＿Ｆｉ

ｌｅｓ＿Ｔｏ＿ＲＡＭを示している。この関数は、図１０の４５６（付属書Ｂの第２５５行）において始まり、まず４５８（付属書Ｂの第２５９行）でＳＧ＿ＴＹＰＥＴＡＢＬＥ構造内になにかＭＩＭＥタイプが定義されているかどうかを判定する。そして、存在していなければ、アイコンが定義されておらず、４６０（付属書Ｂの第２６０行）においてエラーメッセージが返され、４６２（付属書Ｂの第２６１行）で戻りコードが「失敗」に設定されて４６４（付属書Ｂの第２６１～２６２行）においてこの戻りコードが返される。一方、ＳＧ＿ＴＹＰＥＴＡＢＬＥ構造が空でなければ、この関数は、既定のアイコンをＲＡＭにロードし、４６６（付属書Ｂの第２６５行）において、それに対するポインタを設定する。このアイコンポインタの番号は、付属書Ｂの第２６８行のＬＩＮＧＯのｃａｓｔＮｕｍｂｅｒに関連したものになっており、このｃａｓｔＮｕｍｂｅｒを増分するカウンタを第２７０行で設定している。４６８（付属書Ｂの第２７１行）において、ＳＧ＿ＴＹＰＥＴＡＢＬＥ構造内の最初の（次の）エントリを読み込む。そして、４７０（付属書Ｂの第２７２行）において、読み取るエントリが存在していないと判定された場合には、４７２において戻りコードが「成功」に設定され、４６４（付属書Ｂの第２８８行）でその戻りコードが返される。一方、エントリが残っている間は、４７４においてＭＩＭＥタイプが読み込まれ、４７６（付属書Ｂの第２７３行）においてそのＭＩＭＥタイプに対するｉｃｏｎｆｉｌｅｎａｍｅが読み込まれる。そして、４７８（付属書Ｂの第２７４行）において、このＭＩＭＥタイプに関連付けられたアイコンが存在しないと判定されると、４８０（付属書Ｂの第２７５行）において既定のアイコンポインタが割り当てられるが、そうでなければ、４８２（付属書Ｂの第２７７行）において、関数ｒｅａｄ＿ｉｃｏｎｆｉｌｅ（付属書Ｂの第２９５～２９８行）を使用して次のアイコンビットマップとポインタを読み込む。そして、４８４（付属書Ｂの第２７８行）においてアイコンポインタがＮＩＬではないと判定された場合には、第２７９行においてｃａｓｔＮｕｍが増分される。そうでなければ、ｃａｓｔＮｕｍは増分されず、４８０（付属書Ｂの第２８０行）において既定のアイコンポインタが使用されるが、いずれの場合にも、ＭＩＭＥタイプと関連付けられているアイコンポインタが４８６（付属書Ｂの第２８３行）において挿入される。そして、第２８４行でカウンタが増分され、４６８（付属書Ｂの第２８５行）において次のエントリがＳＧ＿ＴＹＰＥＴＡＢＬＥから読み込まれる。

【００５１】

図１１は、特定のＭＩＭＥタイプ又はファイルタイプ用の新しいアイコン及び／又はメッセージハンドラのインストールに使用するｉｎｓｔａｌｌ＿ｔｙｐｅ関数を示している。この関数は、アイコンファイル名、メッセージハンドラプログラム名、及びＭＩＭＥタイプ又はファイルタイプのいずれかを含む入力を取得して６００から始まる（付属書Ｂの第７１～７５行）。この入力は、ユーザーから、或いは、後述する自動手段によって提供することができる。まず、６０２（付属書Ｂの第７８行）においてＴＹＰＥＴＡＢＬＥファイルがＲＡＭに読み込まれ、ＳＧ＿ＴＹＰＥＴＡＢＬＥが作成される。そして、６０４（付属書Ｂの第８０行）での判定において、そのＳＧ＿ＴＹＰＥＴＡＢＬＥが空であれば、６０６（付属書Ｂの第８１～８２行）でエラーが返される。そうでなければ、６０８（付属書Ｂの第８７行）において、６００で指定されたＭＩＭＥタイプ（又はファイルタイプ）をＳＧ＿ＴＹＰＥＴＡＢＬＥから取得する（存在している場合）。ＳＧ＿ＴＹＰＥＴＡＢＬＥにそのＭＩＭＥタイプ（又はファイルタイプ）が存在し、それが既にアイコン及びメッセージハンドラに関連付けられている場合には、６１２（付属書Ｂの第９０行）において、それを再定義するべきかどうかに関してユーザーに確認を求める。そして、ユーザーがＮＯを選択した場合には、６１４（付属書Ｂの第９４行）においてエラーが返される。一方、そのＭＩＭＥタイプが以前に定義されていなかったり、或いはユーザーがそれを再定義することを選択した場合には、６１６（付属書Ｂの第１０２行）において、６００で指定されたメッセージハンドラプログラムの有効性（例：それがハードディスク上又はネットワークに存在するかどうか）をチェックする。そして、それが有効ではない場合には、６１４（付属書Ｂの第１０４～１０５行）においてエラーが返される。一方、メッセージハンドラとＭＩＭＥタイプ（ファイルタイプ）が有効な場合には、６１８（付属書Ｂ

の第１０８行）においてＳＧ＿ＴＹＰＥＴＡＢＬＥ内でそれらを互いに関連付ける。次に、６２０（付属書Ｂの第１１０行）において、６００で指定されたアイコンファイル名の有効性をチェックし、有効でない場合には、６２２（付属書Ｂの第１１２行）で既定のアイコンを指定し、６２４（付属書Ｂの第１１１行）においてエラー警告が返されるが、いずれの場合にも、６２６（付属書Ｂの第１１４行）において、指定されたアイコンファイル名又は（必要に応じて）既定のアイコンがメッセージハンドラ及びＭＩＭＥタイプ（ファイルタイプ）とＳＧ＿ＴＹＰＥＴＡＢＬＥ内で関連付けられる。そして、６２８（付属書Ｂの第１２２～１２５行）でＳＧ＿ＴＹＰＥＴＡＢＬＥからｔｙｐｅｔａｂｌｅ．ｔｘｔファイルに書き戻し、エラーが発生しなければ、６３０（付属書Ｂの第１２６行）において「成功」が返される。

【００５２】

当業者であれば、モジュラーオーサリング／読み取りコンポーネントがインストール／除去される際に、アイコンが自動的にインストール／除去されるよう、ｉｎｓｔａｌｌ＿ｔｙｐｅ関数を別のプログラムによって呼び出すことができることを理解するであろう。本明細書のコード又は図面には具体的に示されていないが、当業者であれば、本明細書の付属書Ｂと親出願のコンポーネントインストールコードを参照することにより、この第２の実施例の実装方法を理解するであろう。

【００５３】

第３の実施例によれば、メールボックスディスプレイヤが新しいメッセージタイプを検出するたびに新しいアイコンが自動的に追加される。これらの新しいアイコンは、オペレーティングシステムのレジストリ又はメッセージに組み込まれたアイコン画像データから取得されるが、当業者であれば、本明細書の付属書Ｂと前述のＭＩＭＥ規格を参照することにより、この実施例の実装方法を理解するであろう。

【００５４】

第４の実施例によれば、必要に応じて、或いはスケジュールに従って、タイプアップデータが新しいアイコン情報についてネットワークサーバーに問い合わせ、アイコン画像データをダウンロードする。当業者であれば、ファイルサーバーからデータをダウンロードする自動アップデータは周知であり、そのような周知の自動アップデータを本明細書の付属書Ｂと共に参照することにより、この実施例を実装できることを理解するであろう。

【００５５】

以上、本明細書では、ＭＩＭＥタイプを識別し異なるアイコンを表示する方法及び装置を含む機能拡張された電子メールシステムのいくつかの実施例について説明及び例証した。尚、本発明の特定の実施例について説明しているが、これは、本発明をそれらに限定することを意図するものではなく、本発明は、その技術が許容するだけの広い範囲を有しており、その仕様も同様に読み取られるべきである。従って、特定のコードリストを開示しているが、その他のコードも利用することができるものと理解されたい。例えば、付属書においては、ＭＡＣＲＯＭＥＤＩＡ　ＤＩＲＥＣＴＯＲのＬＩＮＧＯコードによって本発明を示しているが、本発明は、「ＭＯＺＩＬＬＡ」エンジンに基づいて、Ｃ＋＋により、或いは、その他の数多くの一般的な開発ツールによって実装することができる。又、インストール可能なオーサリング／読み取りコンポーネントを備えるＥメールクライアントと関連する本出願人の親出願を参照して本発明を開示しているが、メールボックスのリストにＭＩＭＥタイプを表すアイコンを表示する本明細書に開示された原理は、その他の種類のＥメールクライアントにも適用可能であることを認識されたい。更には、アイコン画像を保存する方法（即ち、拡大縮尺可能な画像）を参照して特定の構成を開示しているが、その他の構成も同様に使用可能であることを理解されたい。又、本発明はコンピュータ画面上での表示を参照して示されているが、この表示は、テレビ、個人用情報端末、携帯電話、腕時計などにおけるものであってもよいことを理解されたい。従って、当業者であれば、権利請求された精神と範囲を逸脱することなく、本発明に対して更なるその他の変更が可能なことを理解するであろう。

（付属書の簡単な説明）

（付属書Ａ）
本発明によるメールボックスディスプレイヤのＭＣＲＯＭＥＤＩＡ　ＤＩＲＥＣＴＯＲ用のプログラムリストである。
（付属書Ｂ）
本発明によるタイプアップデータのＭＡＣＲＯＭＥＤＩＡ　ＤＩＲＥＣＴＯＲ用のプログラムリストである。
（付属書Ａ：メールボックスディスプレイヤのコード）

```
００１　－－－ メールボックスディスプレイヤの実装
００２　－－－ キッドコード（ＫｉｄＣｏｄｅ）電子メールクライアントのメールボックスハンドラのコード
００３　－－－ このＤｉｒｅｃｔｏｒ ＭＩＡＷは、メールボックスを各メッセージ当たり１行で構成されるリストとしてウィンドウ内に表示するものである。
００４　－－－ 各メッセージは、次の表示フィールドを備えている。
００５　－－－ １．メッセージの番号
００６　－－－ ２．メッセージの送信者
００７　－－－ ３．メッセージのＭＩＭＥタイプとステータス（単一のアイコンを使用して両プロパティを示す）
００８　－－－ ４．メッセージの件名ヘッダ
００９　－－－ ５．メッセージの日付
０１０
０１１
０１２　ｏｎ ｓｔａｒｔＭｏｖｉｅ
０１３　　ｇｌｏｂａｌ ＳＧ＿ｌａｓｔＡｃｔｉｖｅＷｉｎｄｏｗ －－－ ＫＣの最新のアクティブウィンドウを記録しておくスーパーグローバル変数
０１４　　ｇｌｏｂａｌ ｍｂｘＧ＿ｕｓｅｒｎａｍｅ －－－ 現在のユーザー名
０１５　　ｇｌｏｂａｌ ｍｂｘＧ＿ｍｅｓｓａｇｅｓ －－－ メッセージのリスト
０１６　　ｇｌｏｂａｌ ｍｂｘＧ＿ｎＭｓｇｓ －－－ メールボックス内のメッセージの数
０１７　　ｇｌｏｂａｌ ｍｂｘＧ＿ｂｏｘＮａｍｅ －－－ 現在のメールボックス名
０１８　　ｇｌｏｂａｌ ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ －－－ 現在のハイライト行 ＝ ｍｓｇＮｕｍｂｅｒ
０１９　　ｇｌｏｂａｌ ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅ
０２０　　ｇｌｏｂａｌ ｍｂｘＧ＿ｌｉｐｓ
０２１
０２２　　ｓｅｔ ｍｂｘＧ＿ｌｉｐｓ ＝ ０
０２３
０２４　　－－－ ＡＰＩによってメインムービーに呼び出される。
０２５　　ｔｅｌｌ ｔｈｅ ｓｔａｇｅ ｔｏ ｅｍｈ＿ｃｏｎｔｉｎｕｅ（＃ｍａｉｌｂｏｘ）
０２６　ｅｎｄ
０２７
０２８
０２９　ｏｎ ａｃｔｉｖａｔｅＷｉｎｄｏｗ
０３０　　ｇｌｏｂａｌ ＳＧ＿ｌａｓｔＡｃｔｉｖｅＷｉｎｄｏｗ
０３１　　ｇｌｏｂａｌ ｍｂｘＧ＿ｍｙＷｉｎｄｏｗ
０３２　　ｓｅｔ ＳＧ＿ｌａｓｔＡｃｔｉｖｅＷｉｎｄｏｗ ＝ ｍｂｘＧ＿ｍｙＷｉｎｄｏｗ
０３３　ｅｎｄ ａｃｔｉｖａｔｅＷｉｎｄｏｗ
０３４
```

```
035
036   --- ムービーハンドラを停止する。
037   ----------------------------------------
----------------------------------------
----------
038   MIAW内のStopMovieハンドラは、ムービーが最後まで再生されるか、
039   --- 或いは別のムービーにジャンプしたときにのみ呼び出される。
040   --- ウィンドウが閉じられたり、ウィンドウがforgetWindow
041   --- コマンドによって削除された場合には呼び出されない。
042
043   on stopMovie
044       cleanUpMovie()
045   end
046   ----------------------------------------
----------------------------------------
----------
047   MIAWを閉じるか、或いはforgetWindowが呼び出されたときに
048   --- 自動的に呼び出される。
049
050   on closeWindow
051       cleanUpMovie()
052   end
053   ----------------------------------------
----------------------------------------
----------
054
055   --- cleanUpMovieは、stopMovie及びcloseWindowの両方から呼び出す
056   --- ことができる。
057
058   on cleanUpMovie
059       global mbxG_whichLine
060       global mbxG_subtractLine
061       global mbxG_nMsgs
062       global mbxG_username
063       global mbxG_lips
064
065       --- フィールドとグローバル変数をクリアする。
066
067       put "" into field "MailboxTitle"
068       put "" into field "prepositionTitle"
069       put "" into field "MessageNumber"
070       put "" into field "MailboxTo"
071       put "" into field "MailboxSubject"
072       put "" into field "MailboxDate"
073       put "" into field "messageread"
074
075       set mbxG_nMsgs = 0
```

```
076         set mbxG_lips = 0
077         restHilite()
078
079         if findEmpty(member 50) > 50 then
080             set the scriptText of member 50
= ""
081         end if
082
083         set the memberNum of sprite 6 = the
memberNum of member "lips up"
084         set the loc of strite 4 to point(800
, 4)
085         set the loc of sprite 5 = point(800,
19)
086     end cleanUpMovie
087
088
089     --- APIパブリックハンドラ ----------
----------
090     ---
091     --- MIAWのDirector起動に伴う問題に対処するための不恰好な
092     --- 裏技である。その問題とは、MIAWでハンドラを呼び出した後に、
093     --- 呼び出し元のムービーが次のフレームに進むまでMIAW用の
094     --- StartMovieハンドラが実行されないことである。
095     --- 従って、制御が呼び出し元ムービーに返されるまでMIAW内の
096     --- 本当のハンドラが実行されないよう、呼び出し元ムービーの
097     --- 呼び出しシーケンスを設計しなければならない。しかしながら、
098     --- MIAW内の少なくとも1つのハンドラは、StartMovieハン
ドラが実行
099     --- される前に呼び出し元ムービーによって呼び出されなければならない。
100
101     --- startMeupは偽のハンドラであり、メインムービーから呼び出
され
102     --- ると、すぐにメインムービーに戻り、このムービーのstartMov
ie
103     --- ハンドラを実行させる。
104
105     --- この不細工な裏技の第2部分は、MIAWのstartMovieハン
ドラ内
106     --- に含まれている。それは、呼び出し元ムービー内のworkAroun
d
107     --- ハンドラに対するcontinueComponentと呼ばれる呼び
出しである。
108     --- 呼び出し元ムービーのcontinueRebusハンドラがMIAW
内の
109     --- 本当のハンドラを呼び出している。
110
111     on emc_startMeup
112         --- put "Macromedia sucks!"
113         return(TRUE)
```

```
end emc_startMeUp

--------------------------------------------------------------------------------
initWindowは、メッセージハンドラが開かれた際に、Eメールメイン
--- によって呼び出される。

on emc_initWindow userName, windowName
    global mbxG_myWindow
    global mbxG_username
    global mbxG_platformType

--- put "Enter emc_initWindow mailbox"
    set mbxG_username = userName
    set mbxG_myWindow = windowName

    --- puppet hilite (MB 4-17-99) removed this
    resetHilite()
    tell the stage to emh_getColorDepth()
    set colorDepth = the result
    mapColors(colorDepth)
    tell the stage to emh_getPlatformType()
    set mbxG_platformType = the result

--- put "EXIT emc_initWindow Mailbox"

    return(TRUE)

end emc_initWindow

--------------------------------------------------------------------------------
--- closeWindowは、RebusがMIAWとして再生されない限り、呼び出さ
--- れない。

on emc_closeWindow
--- put "ENTER emc_closeWindow Mailbox"
    closeWindow()
    --- step frame
--- put "EXIT emc_closeWindow Mailbox
```

```
”
        return(TRUE)

end emc_closeWhidow

-----------------------------------------------------------------------------------------------

on emc_getComponentInfo
        return( list( ”SimpleMail”, 1, #mailbox, ”text”))
end emc_getComponentInfo

-----------------------------------------------------------------------------------------------

on mbx_getMessage
---
--- --- 「Open」ボタン及びハイライトされているメッセージのダブル
--- --- クリックによってEメールメインが呼び出され、選択された
--- --- メッセージがメッセージ処理ムービーに渡される。
--- --- このスクリプトは、以前は「Open」キャストメンバスクリプトであった。
---
--- global mbxG_whichLine
--- global mbxG_messages
---
--- set mailData = getAt(mbxG_messages, mbxG_whichLine)
---
--- return(mailData)
---
---
---
---end mbx_getMessage

-----------------------------------------------------------------------------------------------

on mbx_getMessageNumber
        global mbxG_whichLine
        return(mbxG_whichLine)
end mbx_getMessageNumber

-----------------------------------------------------------------------------------------------
```

```
－－－－－－－－－－－－－－－－－－－－
１９０　－－－ ｍｂｘ＿ｔｒａｓｈＭｅｓｓａｇｅｓは、メールファイル内の破棄するメッセージの
１９１　－－－ リストを返し、Ｅメールメインが当該メールファイルを書き換える。
１９２　－－－ 正しく実装すれば、メールボックス表示内の現在選択されている
１９３　－－－ 行のメッセージ番号を判定し、表示を更新してそれらのメッセージを
１９４　－－－ リストから削除し、削除されたメッセージ番号のリストを返す。
１９５　－－－
１９６
１９７　ｏｎ ｍｂｘ＿ｔｒａｓｈＭｅｓｓａｇｅｓ
１９８
１９９　　ｇｌｏｂａｌ ｍｂｘＧ＿ｍｅｓｓａｇｅｓ －－－ メッセージのリスト
２００　　ｇｌｏｂａｌ ｍｂｘＧ＿ｎＭｓｇｓ －－－ メールボックス内のメッセージの数
２０１　　ｇｌｏｂａｌ ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ －－－ 現在のハイライト行 ＝ ｍｓｇＮｕｍｂｅｒ
２０２
２０３　　－－－ ｓｅｔ ｍａｉｌＤａｔａ ＝ ｇｅｔＡｔ（ｍｂｘＧ＿ｍｅｓｓａｇｅｓ，ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ）
２０４　　－－－
２０５　　－－－ －－－ ｔｅｌｌ ｔｈｅ ｓｔａｇｅ
２０６　　－－－ ｒｅｔｕｒｎ（ｍａｉｌＤａｔａ）
２０７　　－－－ －－－ ｅｎｄ ｔｅｌｌ
２０８
２０９　　－－－ 複数メッセージを処理可能な実装が必要である。
２１０　　－－－ 又、その他のムービー内で呼び出すことができるのはＡＰＩ
２１１　　－－－ ハンドラのみであるというＡＰＩ規則に準拠していない
２１２　　－－－ ｔｒａｓｈＩｔを書き換えることも必要である。
２１３　　－－－ ｓｅｔ ｍｅｓｓａｇｅ ＝ ｍｂｘＧ＿ｍｅｓｓａｇｅｓ
２１４　　ｉｆ ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ ＞ ０ ＡＮＤ ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ＜＝ ｍｂｘＧ＿ｎＭｓｇｓ ｔｈｅｎ
２１５　　　　ｔｅｌｌ ｔｈｅ ｓｔａｇｅ ｔｏ ｅｍｈ＿ａｌｅｒｔＵｓｅｒＴｏＴｒａｓｈ（）
２１６　　　　ｓｅｔ ｙｅｓ ＝ ｔｈｅ ｒｅｓｕｌｔ
２１７　　　　ｉｆ ｎｏｔ ｙｅｓ ｔｈｅｎ ｒｅｔｕｒｎ［］ －－ ユーザーが取り消される。
２１８
２１９　　　　ｓｅｔ ｔｒａｓｈＬｉｓｔ ＝ ｌｉｓｔ（ｇｅｔＡｔ（ｍｂｘＧ＿ｍｅｓｓａｇｅｓ，ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ））
２２０　　　　ｄｅｌｅｔｅＡｔ（ｍｂｘＧ＿ｍｅｓｓａｇｅｓ，ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ）
２２１　　　　ｓｅｔ ｍｂｘＧ＿ｎＭｓｇｓ ＝ ｍｂｘＧ＿ｎＭｓｇｓ － １
２２２　　　　ｄｉｓｐｌａｙＭａｉｌｂｏｘ（ｍｂｘＧ＿ｍｅｓｓａｇｅｓ）
２２３　　　　ｒｅｓｔＨｉｌｉｔｅ（）
２２４　　ｅｌｓｅ
２２５　　　　ａｌｅｒｔ（”削除するメッセージをクリックしてください。”）
２２６　　　　ｓｅｔ ｔｒａｓｈＬｉｓｔ ＝ ［］
２２７　　ｅｎｄ ｉｆ
２２８
```

```
      return(trashlist)

  end mbx_trashMessages

  ------------------------------------------------------------------------------------------------
  --- ボックス名とメッセージのリストから構成されるメールボックスの
  --- データ構造を受け付ける。

  on mbx_openMailbox mailbox
      global mbxG_username
      global mbxG_messages
      global mbxG_boxName
      global mbxG_nMsgs

  --- put "ENTER mbx_openMailbox"
      set mbxG_boxName = getAt(mailbox, 1)
      put mbxG_userName & "'s" & mbxG_boxName into field "mailboxTitle"

      set bmxG_messages = getAt(mailbox, 2)
      set mbxG_nMsgs = count(mbxG_messages)

      displayMailbox(mbxG_messages)

  --- put "EXIT mbx_openMailbox"
      return(TRUE)

  end mbx_openMailbox

  ------------------------------------------------------------------------------------------------

  Utilities --
  ------------------------------------------------------------------------------------------------
  テキストフィールドの書式設定を初期化する。
  --- これについてはFrank Leahyに感謝する。

  on SetTextInfo fldName, FldValue, FldAlign, FldFont, FldSize, FldStyle
      if fldValue <> EMPTY then
          put fldValue into field fldName
      end if
```

```
268        set the textAlign of field fldName
= fldAlign
269        set the textFont of field fldName =
fldFont
270        set the textSize of field fldName =
fldSize
271        set the textStyle of field fldName
= fldStyle
272    end
273
274    ------------------------------------
------------------------------------
-------------------
275    on formatFields
276
277        --- テキストフィールドの書式設定を行う。
278
279        setTextInfo "MessageNumber", " ", "
left", "arial", 14, "bold"
280        setTextInfo "MailboxTo", " ", "left
", "arial", 14, "bold"
281        setTextInfo "MailboxSubject", " ",
"left", "arial", 14, "bold"
282        setTextInfo "MailboxDate", " ", "le
ft", "arial", 14, "bold"
283        setTextInfo "Messageread", " ", "le
ft", "arial", 14, "bold"
284
285    end formatFields
286
287    --- メインメールボックス表示関数
288    --- メールボックススタイルのメッセージリストを表示する。
289    --- 各メッセージの適切な構成要素をフィールドメンバ内に
290    --- 配置し、各行を揃えて表示する。
291
292    on displayMailBox msgList
293        global SG_TYPETABLE --- 様々なMIAWで共有
するスーパーグローバル変数
294        global mbxG_red
295        global mbxG_platformType
296        global mbxG_iconList
297
298        --- mbxG_iconListは、上／下スクロールスクリプトで
将来使用する。
299        set mbxG_iconList = [:]
300        set count = 0
301
302        --- まず、すべてのフィールドとスプライトをクリアする。
303        put "" into field "MessageNumber"
304        put "" into field "MailboxTo"
```

```
305         put "" into field "mailboxSubject"
306         put "" into field "mailboxDate"
307         put "" into field "Messageread"
308
309         repeat with i = 40 to 70
310             set the member of sprite i = memb
er "blank"
311         end repeat
312
313         --- ウィンドウのスクロールの際に供給されるよう、すべての
314         --- メッセージの情報をテキストフィールドに書き込む。
315         repeat with msg in msgList
316
317         --- メールボックス、MIMEタイプ、及びステータスを除き、
318         --- 書き込まれるとそのフィールドは自動的に表示される。
319
320         put the lineCount of member "Mailbo
xTo"+1 & RETURN after field "MessageNumber
"
321         put getProp(msg, #from) & RETURN af
ter field "MailboxTo"
322         put getProp(msg, #subject) & RETUR
N after field "mailboxSubject"
323         put getProp(msg, #date) & RETURN af
ter field "mailboxDate"
324         put getProp(msg, #mailbox) into ma
ilbox
325         put getProp(msg, #mimetype) into m
ime
326         put getProp(msg, #status) into sta
tus
327
328         ------- MIMEタイプのアイコンを表示する。
329         ------- メッセージステータスの表示にもMIMEタイプのアイコ
ンを使用している。
330         --- メッセージが既読の場合には、グレースケールバージョンのアイ
コンを表示する。既読でなければ、カラーのアイコンを表示する。
331
332         ------- アイコンは、40番以上のスプライトチャネルに配置する
。
333             set i = 40 + count
334             set count = count + 1
335
336         --- メッセージのMIMEタイプ用のアイコンを見つけ出す。
337         ------- このコードはデータアクセス関数を使用して書き換えるべ
きである。
338         ------- 現時点では、TYPETABLEデータ構造のフォーマッ
トを知る必要がある。
339
340             set mimeProperties = getProp(SG
```

```
_ＴＹＰＥＴＡＢＬＥ，　ｍｉｍｅ）
３４１　　ｓｅｔ　ｉｃｏｎＣａｓｔＭｅｍｂｅｒ　＝　ｇｅｔＡｔ（ｍｉｍｅ
Ｐｒｏｐｅｒｔｉｅｓ，　２）
３４２　　－－－　２番目の項目はキャストメンバ番号である。
３４３
３４４　　－－－　このタイプが未知のものであれば、既定のアイコンを使用する。
３４５　　ｉｆ　ｉｃｏｎＣａｓｔＭｅｍｂｅｒ　＝　０　ｔｈｅｎ　ｓｅｔ　ｉ
ｃｏｎＣａｓｔＭｅｍｂｅｒ　＝　ｔｈｅ　ｎｕｍｂｅｒ　ｏｆ　ｍｅｍｂｅｒ
３４６　　”ＤｅｆａｕｌｔＩｃｏｎ”
３４７
３４８　　－－－　メッセージが既読の場合には、グレースケールバージョンのアイ
コンを取得する。
３４９　　ｉｆ　ｓｔａｔｕｓ　＝　”Ｒ”　ｔｈｅｎ　ｓｅｔ　ｉｃｏｎＣａｓ
ｔＭｅｍｂｅｒ　＝　ｉｃｏｎＣａｓｔＭｅｍｂｅｒ　＋　１
３５０
３５１　　－－－　このアイコンをウィンドウのスクロールに使用するアイコンリス
トに追加する。
３５２　　ａｐｐｅｎｄ　（ｍｂｘ＿ｉｃｏｎＬｉｓｔ，　ｉｃｏｎＣａｓｔ
Ｍｅｍｂｅｒ）
３５３
３５４　　－－－　表示用の適切な位置にこのアイコンを配置する。
３５５　　ｓｅｔ　ｔｈｅ　ｍｅｍｂｅｒＮｕｍ　ｏｆ　ｓｐｒｉｔｅ　ｉ　ｔｏ
ｉｃｏｎＣａｓｔＭｅｍｂｅｒ
３５６
３５７　　－－－　このアイコンをメールボックスのメッセージリスト内の正しいグ
リッドセルに表示する。
３５８　　－－－　すべてのアイコンのｌｏｃＨを５０に設定する。
３５９　　ｓｅｔ　ｗｈｅｒｅＧｏｅｓＩｃｏｎ　＝　ｔｈｅ　ｌｉｎｅＣｏｕｎｔ　ｏ
ｆ　ｍｅｍｂｅｒ　”ＭａｉｌｂｏｘＴｏ”
３６０　　ｐｕｐｐｅｔＳｐｒｉｔｅ　ｉ，　Ｔｒｕｅ
３６１　　ｓｅｔ　ｔｈｅ　ｖｉｓｉｂｌｅ　ｏｆ　ｓｐｒｉｔｅ　ｉ　＝　ＴＲＵＥ
３６２　　ｓｅｔ　ｔｈｅ　ｌｏｃＨ　ｏｆ　ｓｐｒｉｔｅ　ｉ　ｔｏ　５０
３６３
３６４　　－－－　メッセージの隣にアイコンを正確に配置する。
３６５　　ｓｅｔ　ｐｏｓｉｔｏｎＶａｒ　＝　１０５　＋　ｌｉｎｅＰｏｓＴｏＬｏｃ
Ｖ（ｍｅｍｂｅｒ　”ＭａｉｌｂｏｘＴｏ”，　ｗｈｅｒｅＧｏｅｓＩｃｏｎ）
３６６　　ｓｅｔ　ｔｈｅ　ｌｏｃＶ　ｏｆ　ｓｐｒｉｔｅ　ｉ　ｔｏ　ｐｏｓｉｔｏｎＶ
ａｒ
３６７　　－－－　但し、多数のメッセージが存在する場合、アイコンがウィンドウ
からはみ出さないようにする。
３６８　　ｉｆ　ｐｏｓｉｔｏｎＶａｒ　＞　５５０　ｏｒ　ｐｏｓｉｔｏｎＶａｒ　＜
１０５　ｔｈｅｎ
３６９　　ｓｅｔ　ｔｈｅ　ｖｉｓｉｂｌｅ　ｏｆ　ｓｐｒｉｔｅ　ｉ　＝　ＦＡＬ
ＳＥ
３７０　　ｅｌｓｅ
３７１　　ｓｅｔ　ｔｈｅ　ｖｉｓｉｂｌｅ　ｏｆ　ｓｐｒｉｔｅ　ｉ　＝　ＴＲＵ
Ｅ
３７２　　ｅｎｄ　ｉｆ
３７３　　ａｄｄＰｒｏｐ　ｍｂｘＧ＿ｉｃｏｎＬｉｓｔ，　（ｔｈｅ　ｌｏｃＶ　ｏ
ｆ　ｓｐｒｉｔｅ　ｉ），　ｍｉｍｅ
```

```
374         end repeat
375
376     end displayMailbox
377
378     --- ユーザーのメールボックスとのやり取りへの応答に使用する関数
379     --- HILITE MESSAGEは、ユーザーがメッセージの行をマウスでクリックすると呼び出される。
380
381     on hiliteMessage whichLine
382         global mbxG_nMsgs, mbxG_whichLine, mbxG_subtractLine, mbxG_messages
383
384     --- 選択された行を記録する。
385
386     set mbxG_whichLine = whichLine
387
388     --- 行が有効であることを確認する
389
390     if mbxG_whichLine <= 0 then
391         return(0) --- なにもしない。エラーは別のところで取得される。
392     else if mbxG_whichLine > mbxG_nMsgs then
393         --- ユーザーはフィールド内の別の部分をクリックした。
394         set mbxG_whichLine = 0 --- 0にリセットする。
395         return(0)
396     end if
397
398     --- 選択された行をハイライトする。
399
400     set whichHighlight = mbxG_whichLine + mbxG_subtractLine
401
402     --- 表示されているすべてのフィールドメンバが同期状態に維持されて
403     --- いるため、どれでもlinePosToLocVに使用できる。
404     --- 小さいため、"MailboxTo"を使用する。
405
406     set the locV of Sprite 11 to -
407         (99 + linePosToLocV(member "MailboxTo", whichHighlight))
408
409     --- ハイライトされたメッセージをユーザーがダブルクリックした際に、mailFileからそのメッセージを取得する。
410
411     if the doubleClick then
412         set maildata = getAt(mbxG_message, mbxG_whichLine)
413         tell the stage
414             emh_openMessage(maildata)
```

```
415         end tell
416         --- mbx_getMessage()
417     end if
418
419     end hiliteMessage
420
421
422     ------------------------------------------------------------
------------------------------------------------------------
423     on restHilite
424         global mbxG_whichLine, mbx_subtra
ctLine
425
426         set mbxG_whichLine = 0
427         set mbxG_subtractLine = 0
428
429         --- ハイライトをオフステージに設定する。
430         set the loc of sprite 11 to point(11
, -20)
431
432     end resetHilite
433     ------------------------------------------------------------
------------------------------------------------------------
434     --- これは、色のルックアップテーブルである。
435     --- これを実際に必要とするのは、Macプラットフォームにおいてのみで
ある。
436     --- フィールドの前景色に設定したい色の上で使用する。
437     ---
438
439     on mapColors colorDepth
440         global mbxG_red
441         global mbxG_blue
442         global mbxG_white
443         global mbaG_black
444
445         case colorDepth of
446
447             8:
448                 set mbxG_red = 6
449                 set mbxG_blue = 4
450                 set mbxG_white = 0
451             16:
452                 set mbxG_red = 31744
453                 set mbxG_blue = 31
454                 set mbxG_white = 32767
455             32:
456                 set mbxG_red = 16711680
457                 set mbxG_blue = 255
```

```
458                 set mbxG_white = 16777215
459
460         end case
461
462         set mbxG_black = the forecolor of l
ine 1 of member the member of sprite 4
463
464     end mapColors
465
466     ------------------------------------------
------------------------------------------
------------------
```

467 メールボックスのメッセージの行をマウスでクリックすると、スクリプトが実行される。

468 メッセージ行のフィールドごとにスクリプトが必要である。

```
469
470     on mouseUp
471         hiliteMessage (the clickon - 40)
472     end
473
474
475     on mouseDown
476
477         global mbxG_lips
478         set whichLine = the mouseLine
479         if mbxG_lips then
480             set astr = line whichline of fie
ld "MailboxTo"
481             speak(astr)
482         else
483             hiliteMessage(whichLine)
484         end if
485     end
486
487     on mouseDown
488
489         global mbxG_lips
490         set whichLine = the mouseLine
491         if mbxG_lips then
492             set astr = line whichline of fie
ld "MailboxSubject"
493             speak(astr)
494         else
495             hiliteMessage(whichLine)
496         end if
497     end
498
499     ------------------------------------------
------------------------------------------
------------------
```

```
５００　メールボックスのメッセージ行をマウスでクリックすると、更なるスクリプトが実行される。
５０１
５０２　ｏｎ ｍｏｕｓｅＤｏｗｎ
５０３　　ｇｌｏｂａｌ ｍｂｘＧ＿ｌｉｐｓ
５０４　　ｓｅｔ ｗｈｉｃｈＬｉｎｅ ＝ ｔｈｅ ｍｏｕｓｅＬｉｎｅ
５０５　　ｉｆ ｍｂｘＧ＿ｌｉｐｓ ｔｈｅｎ
５０６
５０７　　　ｓｅｔ ａｓｔｒ ＝ ｌｉｎｅ ｗｈｉｃｈｌｉｎｅ ｏｆ ｆｉｅｌｄ ”ＭａｉｌｂｏｘＤａｔｅ”
５０８　　　ｓｅｔ ａｄａｙ ＝ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５０９　　　ｃａｓｅ ａｄａｙ ｏｆ
５１０　　　　”Ｍｏｎ，”：ｐｕｔ ”Ｍｏｎｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１１　　　　”Ｔｕｅ，”：ｐｕｔ ”Ｔｕｅｓｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１２　　　　”Ｗｅｄ，”：ｐｕｔ ”Ｗｅｄｎｅｓｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１３　　　　”Ｔｈｕ，”：ｐｕｔ ”Ｔｈｕｒｓｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１４　　　　”Ｆｒｉ，”：ｐｕｔ ”Ｆｒｉｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１５　　　　”Ｓａｔ，”：ｐｕｔ ”Ｓａｔｕｒｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１６　　　　”Ｓｕｎ，”：ｐｕｔ ”Ｓｕｎｄａｙ” ｉｎｔｏ ｗｏｒｄ １ ｏｆ ａｓｔｒ
５１７　　　　ｏｔｈｅｒｗｉｓｅ
５１８　　　ｅｎｄ ｃａｓｅ
５１９
５２０　　　ｓｐｅａｋ（ａｓｔｒ）
５２１　　ｅｌｓｅ
５２２　　　ｈｉｌｉｔｅＭｅｓｓａｇｅ（ｗｈｉｃｈＬｉｎｅ）
５２３　　ｅｎｄ ｉｆ
５２４　ｅｎｄ
５２５
５２６　ｏｎ ｍｏｕｓｅＤｏｗｎ
５２７　　ｇｌｏｂａｌ ｍｂｘＧ＿ｌｉｐｓ
５２８
５２９　　ｓｅｔ ｗｈｉｃｈＬｉｎｅ ＝ ｔｈｅ ｍｏｕｓｅＬｉｎｅ
５３０　　ｉｆ ｍｂｘＧ＿ｌｉｐｓ ｔｈｅｎ
５３１　　　ｓｅｔ ａｓｔｒ ＝ ｌｉｎｅ ｗｈｉｃｈｌｉｎｅ ｏｆ ｆｉｅｌｄ ”ＭｅｓｓａｇｅＮｕｍｂｅｒ”
５３２　　　ｓｐｅａｋ（ａｓｔｒ）
５３３　　ｅｌｓｅ
５３４　　　ｈｉｌｉｔｅＭｅｓｓａｇｅ（ｗｈｉｃｈＬｉｎｅ）
５３５　　ｅｎｄ ｉｆ
５３６　ｅｎｄ
５３７
５３８
５３９　－－－ このスクリプトは、メッセージ行に表示されるメッセージタイプアイコ
```

```
ンに付加される。
５４０
５４１   ｏｎ ｍｏｕｓｅＵｐ
５４２       ｓｅｔ ｗｈｉｃｈＬｉｎｅ ＝ ｔｈｅ ｍｏｕｓｅＬｉｎｅ
５４３       ｈｉｌｉｔｅＭｅｓｓａｇｅ（ｍｏｕｓｅＬｉｎｅ）
５４４   ｅｎｄ
５４５
５４６   －－－－－－－ スクロールボタン用のコード
５４７
５４８   ｏｎ ｍｏｕｓｅＤｏｗｎ
５４９       ｇｌｏｂａｌ ｍｂｘＧ＿ｗｈｉｃｈＬｉｎｅ
５５０       ｇｌｏｂａｌ ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅ
５５１       ｇｌｏｂａｌ ｍｂｘＧ＿ｉｃｏｎＬｉｓｔ
５５２
５５３       －－－ ハイライトされた状態で上にスクロールする。
５５４       －－－ これでアイコンはスクロールする。しかし、アイコンの
５５５       －－－ スプライト位置は、ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅに基づ
いており、
５５６       －－－ メールボックスウィンドウ内のメッセージの対応する実際の
５５７       －－－ 行番号には基づいていない。
５５８
５５９       ｓｅｔ ｎｕｍｂｅｒＯｆＩｃｏｎｓＶａｒ ＝ ｃｏｕｎｔ（ｍｂｘＧ＿
ｉｃｏｎＬｉｓｔ）
５６０       ｓｅｔ ｌａｓｔＩｃｏｎＰｏｓ ＝ ｇｅｔＰｒｏｐＡｔ（ｍｂｘＧ＿ｉ
ｃｏｎＬｉｓｔ，ｃｏｕｎｔ（ｍｂｘＧ＿ｉｃｏｎＬｉｓｔ）
５６１       ｉｆ ｌａｓｔＩｃｏｎＰｏｓ ＞＝ ５５０ ｔｈｅｎ
５６２           －－－ メッセージが多数存在しており、スクロールが必要である
。
５６３           ｒｅｐｅａｔ ｗｈｉｌｅ ｔｈｅ ｍｏｕｓｅＤｏｗｎ ＝ ＴＲＵ
Ｅ
５６４
５６５           －－－ ｏｌｄＳｕｂｔｒａｃｔＬｉｎｅは、ｍｂｘＧ＿ｓｕｂｔ
ｒａｃｔＬｉｎｅを増分する
５６６           －－－ 前にｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅに設定する。こ
れにより、
５６７           －－－ 対象メッセージより１つ後ろにアイコンがずれないように
している。
５６８
５６９           ｓｅｔ ｏｌｄＳｕｂｔｒａｃｔＬｉｎｅ ＝ ｍｂｘＧ＿ｓｕｂｔ
ｒａｃｔＬｉｎｅ
５７０           ｓｅｔ ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅ ＝ ｍｂｘＧ＿ｓｕ
ｂｔｒａｃｔＬｉｎｅ ＋ １
５７１
５７２           ｉｆ ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅ ＞ ０ ｔｈｅｎ
５７３               ｓｅｔ ｍｂｘＧ＿ｓｕｂｔｒａｃｔＬｉｎｅ ＝ ０
５７４           ｅｎｄ ｉｆ
５７５
５７６       －－－ すべてのフィールドを一緒にスクロールする。
５７７
５７８       ｓｃｒｏｌｌＢｙＬｉｎｅ ｍｅｍｂｅｒ ”ＭｅｓｓａｇｅＮｕｍｂｅ
```

```
r", −1
579         scrollByLine member "MailboxTo", −
1
580         scrollByLine member "MailboxSubje
ct", −1
581         scrollByLine member "MailboxDate"
, −1
582         scrollByLine member "mime", −1
583         scrollByLine member "Messageread"
, −1
584
585         set numberOfIcons = the lineCount o
f member "MailboxTo" + 40586        set amount
OfMail = the lineCount of member "MailboxTo
"
587
588         −−− デバック用
589         −−− put "linecount:" & the lineCoun
t of member "MailboxTo"
590         −−− put "subtractline:" & mbxG_sub
tractLine
591
592         repeat with i = 40 to numberOfIcon
s
593
594
595             −−− 次の2条件が真であれば、アイコンをスクロールする。
596             −−− 即ち、メッセージがスクロールすれば、アイコンもスクロ
ールし、メッセージがスクロールしなければ、アイコンもスクロールしない。597
598             if mbxG_subtractLine >= − amoun
tOfMail + 1 and oldSubtractLine < 0 then
599                 set the locV of sprite i = the
 locV of sprite i + 15
600                 if the locV of sprite i < 105
then
601                     set the visible of sprite
i = FALSE
602                 else
603                     set the visible of sprite
i = TRUE
604                 end if
605             else
606                 nothing
607             end if
608         end repeat
609
610         −−− 行と共にハイライトを移動させ、行が画面から外れたら、
611         −−− ハイライトも画面から外す。
612         set whichHighlight = mbxG_whichLin
e + mbxG_subtractLine
```

```
613         if whichHighlight <= 0 or whichHigh
light >= 22 then
614         set the loc of sprite 11 to point(11
, -20)
615         else --- スクロールされたメッセージにハイライトのlocV
を設定する。
616         set the locV of Sprite 11 to (99 + li
nePosToLocV(member "MailboxTo", whichHigh
light))
617         end if
618         updateStage
619         end repeat
620         end if
621   end
622
623   on mouseUp
624         set numberOfIcons = the lineCount o
f member "MailboxTo" + 40
625         repeat with i = 40 to numberOfIcons
626             if the locV of sprite i > 550 or t
he lobV of sprite i< 105 then
627                 set the visible of sprite i =
FALSE
628             else
629                 set the visible of sprite i =
TRUE
630             end if
631         end repeat
632   end
633
634   on mouseDown
635         global mbxG_whichLine
636         global mbxG_subtractLine
637         global mbxG_iconList
638
639
640         --- メッセージがスクロールする際には、
641         アイコンもメッセージと共に移動する必要があり、
642         スクロールした後に表示される新しい
643         --- メッセージに
644         アイコンスプライトのmemberNumを
645         --- 割り当てる。
646         --- これでアイコンがスクロールする。しかし、
647         --- アイコンスプライトの位置は、mbxG_subtractLi
ne
648         --- に基づいており、メールボックスウィンドウ内のメッセージの
649         --- 対応する実際の行番号には基づいていない。
650         set numberOfIconsVar = count(mbxG
_iconList)
651         set lastIconPos = getPropAt(mbxG_i
```

```
conList, Count(mbxG_iconList)
652       if lastIconPos >= 550 then
653           --- 多数のメッセージが存在しており、スクロールが必要である。
654           repeat while the mouseDown = TRUE
655               scrollByLine member "MessageNumber", 1
656               scrollByLine member "MailboxTo", 1
657               scrollByLine member "MailboxSubject", 1
658               scrollByLine member "MailboxDate", 1
659               scrollByLine member "mime", 1
660               scrollByLine member "Messageread", 1
661               --- mouseLineとlinePosTolocVの間の
662               違いを補正するために
663               --- 使用する数値を取得する。
664               --- mouseLineはフィールド合計内の行を与える。
665               --- linePosTolocVは画面上のフィールドの行を使用している。
666               set bmxG_subtractLine = mbxG_subtractLine - 1
667               set numberOfIcons = the lineCount of member "MailboxTo" + 40
668               set amountOfMail = the lineCount of member "MailboxTo"
669               --- put "linecount:" & the lineCount of member "MailboxTo"
670               --- put "subtractline:" & mbxG_subtractLine
671               repeat with i = 40 to numberOfIcons
672                   if mbxG_subtractLine >= -amountOfMail + 1 then
673                       set the locV of sprite i = the locV of sprite i - 15
674                       if the locV of sprite i < 105 then
675                           set the visible of sprite i = FALSE
676                       else
677                           set the visible of sprite i = TRUE
678                       end if
```

```
679                     else
680                         nothing
681                     end if
682                 end repeat
683
684                 if mbxG_subtractLine < - amo
untOfMail + 1 then
685                     set mbxG_subtractLine = -
 amountOfMail + 1
686                 end if                                  10
687
688                 --- ハイライトを行と共に移動させ、行が画面から
689                 --- 外れたら、ハイライトも画面から外す。
690
691                 set whichHighlight = mbxG_wh
ichLine + mbxG_subtractLine
692                 if whichHighlight <= 0 or whi
chHighlight >= 22 then
693                     set the loc of sprite 11 t
o point (11, -20)                                       20
694                 else
695                     set the locV of Sprite 11
to (99 +
696                     linePosToLocV(member "Ma
ilboxTo", whichHighlight))
697                 end if
698                 updateStage
699             end repeat
700         end if
701     end                                                 30
702
703     ------
704
705     on emc_indicateCheckingInternet
706         global mbxG_red
707         global mbxG_blue
708         global mbxG_white
709         global mbxG_black
710
711         if the locH of sprite 4 > 600 then              40
712             set the loc of sprite 4 = point(2
23, 4)
713         end if
714         --- if the locH of sprite 5 > 600 the
n
715         ---  set the loc of sprite 5 = point
(509, 19)
716         --- end if
717
718         set colorNow = the forecolor of lin             50
```

```
e 1 of member the member of sprite 4
719
720       case colorNow of
721           mbxG_black: set colorNext = mbx
G_blue --- 青
722           mbxG_blue: set colorNext = mbxG
_white --- ピンク
723           mbxG_white: set colorNext = mbx
G_red --- 赤
724           mbxG_red: set colorNext = mbxG_
blue --- 青
725       end case
726
727       set the forecolor of line 1 of membe
r the member of sprite 4 to colorNext
728       updateStage
729
730   end emc_indicateCheckingInternet
731
732   on emc_endIndicateCheckingInternet
733       set the loc of sprite 4 to point(800
, 4)
734       set the loc of sprite 5 = point(800,
 19)
735       cursor - 1
736       updateStage
737   end emc_endIndicateCheckingInternet
738
739   ---------- ウィンドウクローズボタン用のスクリプト
740
741   on mouseDown
742
743       repeat while the stillDown
744           if inside(point(the mouseH, the
 mouseV), the rect of sprite the clickon) th
en
745               if the name of member the mem
ber of sprite the clickon = "closeWindow"
746               then
747                   set the member of sprite t
he clickon = "closeWindow_down"
748                   updateStage
749               end if
750           else
751               set the member of sprite the
clickon = "closeWindow"
752               updateStage
753           end if
754       end repeat
755       set the member of sprite the clicko
```

```
n = "closeWindow"
756         updateStage
757
758     end mouseDown
759
760     on mouseUp
761
762         --- ウィンドウを閉じる。
763         if inside(point(the mouseH, the mo
useV), the rect of sprite the clickon) then
764             --- 以下の2行は、クローズに当たってメールボックス
765             --- アイコンの処理をスピードアップするためのものである。
766             --- 低速のマシンでこれをチェックする必要がある。
767             hideMailIcons(the lineCount of
member "MailboxTo")
768             go frame "stop"
769             tell the stage to emh_killCompo
nent(0, "")
770             set success = the result
771             if success <> TRUE then
772                 alert("メールボックスＭＩＡＷを閉じる際にエラ
ーが発生しました。")
773             end if
774         end if
775     end
776
777     on hideMailIcons numberOfIcons
778         repeat with i = 40 to (40 + numberOf
Icons)
779             set the visible of sprite i = FAL
SE
780         end repeat
781         updateStage
782     end
783
784
785     --- オープンボタン用のスクリプト
786
787     on mouseDown
788
789         repeat while the stillDown
790             if inside(point(the mouseH, the
 mouseV), the rect of sprite the clickon) th
en
791                 if the name of member the mem
ber of sprite the clickon = "open" then
792                     set the member of sprite t
he clickon = "open_down"
793                     updateStage
794                 end if
```

```
795             else
796                 set the member of sprite the
clickon = ”open”
797                 updateStage
798             end if
799         end repeat
800         set the member of sprite the clicko
n = ”open”
801         updateStage
802     end mouseDown
803
804
805     on mouseUp
806
807         if inside(point(the mouseH, the mo
useV), the rect of sprite the clickon) then
808             global mbxG_whichLine, mbxG_me
ssages
809             if mbxG_whichLine = 0 then
810                 alert ”マウスでクリックし、メッセージを選択して
ください。”)
811                 exit
812             end if
813             set maildata = getAt(mbxG_messa
ges, mbxG_whichLine)
814             tell the stage
815                 emh_openMessage(maildata)
816             end tell
817         end if
818     end
```

（付属書Ｂ：タイプアップデータのコード）

```
001   --- タイプアップデータの実装
002   --- キッドコード（KidCode）電子メールクライアントのメッセージ
タイプ情報を
003   --- 維持管理するコンポーネントのコードである。
004   --- このDirector MIAWにより、キッドコードのその他のコン
ポーネント
005   から呼び出し可能なパブリック関数が提供される。
006
007   --- パブリック関数
008   --- 1. Initialize_TypeTable
009   --- 2. Install_Type
010   --- 3. Uninstall_Type
011
012
013
014   --- 内部でのみ使用されるプライベート関数
015   --- 1. Write_Typetable_File
016   --- 2. Read_Typetable_File
017   --- 3. Read_Icon_Files_To_RAM
```

```
018   --- 4. read_iconFile
019   --- 5. delete_mimetype
020   --- 6. insert_mimetype
021   --- 7. delete_filetype
022   --- 8. insert_filetype
023
024
025   --- TYPETABLEファイルの永久的な保存バージョンのファイル名は
026   --- typetable.txtであり、既定のディレクトリはcurre
ntPathディレクトリである。
027
028
029   --- Initialize_TypeTableは、MIMEタイプアイコ
ン、添付ファイルタイプアイコン、
030   及びメッセージハンドラMIAWをルックアップするのに使用されるデータ構
造を初期化する。
031   --- 使用する前に、まずSG_Typetableをセットアップする。
032   ファイル添付情報は、添付ファイルを有するメッセージを検出した際に使用す
るまでルックアップしない。
033
034   on Initialize_TypeTable
035   global SG_TYPETABLE --- 様々なMIAWで共有するT
ypetable用
036   のスーパーグローバル変数である。
037   global SG_ATTACH_TYPETABLE --- 添付ファイル
タイプ情報を維持管理する。
038
039   set SG_TYPETABLE = [:]     --- MIME情報のプロ
パティリストを初期化する。
040
041   --- ファイルタイプ情報のプロパティリストを初期化する。
042   --- このリストは、添付ファイルを有するメッセージを検出したときにのみ
記入される。
043   set SG_ATTACH_TYPETABLE = [:]
044   set SG_TYPETABLE = Read_TypeTable_Fil
e(the pathname & "typetable.txt")
045
046   if count(SG_TYPETABLE) = 0 then   --- t
ypetableファイルの読み取りに失敗した。
047       alert("エラー:MIMEタイプファイルの読み取りに失敗しま
した。")
048       return(0)
049   end if
050
051   set retVal = Read_Icon_Files_To_RAM()
052
053   if retVal = 0 then
054       alert("エラー:MIMEタイプアイコンのロードに失敗しまし
た。")
055       return(0)
```

```
０５６　ｅｌｓｅ
０５７　　ｒｅｔｕｒｎ（１）
０５８　ｅｎｄ ｉｆ
０５９
０６０　ｅｎｄ　－－－ Ｉｎｉｔｉａｌｉｚｅ＿ＴｙｐｅＴａｂｌｅ
０６１
０６２
０６３
０６４　－－－ Ｉｎｓｔａｌｌ＿Ｔｙｐｅは、システムに新しいＭＩＭＥタイプをインストールするために使用する。
０６５　Ｉｎｓｔａｌｌ＿Ｔｙｐｅは、ＭＩＭＥタイプ（文字列）、メッセージ処理ムービーのファイル名、
０６６　ＭＩＭＥタイプアイコンを格納するビットマップのファイル名、及び任意選択でファイル拡張子（文字列）を入力として取得する。
０６７　この関数は、ＭＩＭＥタイプに関連する情報（関数パラメータとして与えられたもの）を
０６８　永久ストレージに記録されるＭＩＭＥタイプテーブルに追加する。
０６９　ここでは、ＭＩＭＥタイプ情報の永久ストレージとして「ｔｙｐｅｔａｂｌｅ．ｔｘｔ」というファイルを使用している。
０７０
０７１　ｏｎ Ｉｎｓｔａｌｌ＿Ｔｙｐｅ ｍｉｍｅＴｏＩｎｓｔａｌｌ，ｍｓｇＨａｎｄｌｅｒ＿ｆｉｌｅｎａｍｅ，ｉｃｏｎ＿ｆｉｌｅｎａｍｅ，ｆｉｌｅｔｙｐｅ
０７２　ｇｌｏｂａｌ ＳＧ＿ＴＹＰＥＴＡＢＬＥ －－－ インストールされているすべてのＭＩＭＥタイプに関する情報
０７３
０７４　ｓｅｔ ＤＥＦＡＵＬＴ＿ＩＣＯＮＦＩＬＥ ＝ ”ｄｅｆａｕｔＩｃｏｎ．ｂｍｐ”
０７５　ｓｅｔ ＳＧ＿ＴＹＰＥＴＡＢＬＥ ＝ ［：］　－－－ ＭＩＭＥタイプ情報のプロパティリストを初期化する。
０７６
０７７　－－－ 既存のＭＩＭＥタイプ情報をＲＡＭに読み込む。
０７８　ｓｅｔ ＳＧ＿ＴＹＰＥＴＡＢＬＥ ＝ Ｒｅａｄ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅ（ｔｈｅ ｐａｔｈｎａｍｅ ＆ ”ｔｙｐｅｔａｂｌｅ．ｔｘｔ”）
０７９
０８０　ｉｆ ｃｏｕｎｔ（ＳＧ＿ＴＹＰＥＴＡＢＬＥ） ＝ ０ ｔｈｅｎ　－－－ ｔｙｐｅｔａｂｌｅファイルの読み取りに失敗した。
０８１　　ａｌｅｒｔ（”エラー：ＭＩＭＥタイプファイルの読み取りに失敗しました。”）
０８２　　ｒｅｔｕｒｎ（０）
０８３　ｅｎｄ ｉｆ
０８４
０８５　－－－ ＭＩＭＥタイプが既にインストールされているかどうかをチェックする。
０８６
０８７　ｓｅｔ ｍｉｍｅＰｒｏｐｅｒｔｉｅｓ ＝ ｇｅｔ＿ｍｉｍｅｔｙｐｅ（ｍｉｍｅＴｏＩｎｓｔａｌｌ）
０８８
０８９　ｉｆ ｍｉｍｅＰｒｏｐｅｒｔｉｅｓ ＜＞ ０ ｔｈｅｎ－－－ ＭＩＭＥタイプは既にインストールされている。
０９０　　ｓｅｔ ｒｅｄｅｆｉｎｅＡｌｅｒｔ ＝ ｂａＭｓｇＢｏｘ（ｔｈｅＭｅ
```

```
ssage, "KidCode", "YesNoCancel", "Question", 1)
091
092   --- 警告関数はメッセージを保存せず、警告するだけである。
093       case redefineAlert of
094           "No": return 0
095           "Cancel": return n1
096           otherwise: nothing --- 継続
097       end case
098   end if
099
100   --- 新しいMIMEタイプを定義する。
101
102   if verifyMessageHandler(msgHandler_f
ilename) = 0 then  --- プログラムファイルに
103   問題が存在している。
104       alert("エラー：メッセージハンドラプログラム" && msg
Handler_filename "が無効です。")
105       return (0)
106   end if
107
108   insert_msgHandler(SG_TYPETABLE, mime
ToInstall, msgHandler_filename)
109
110   if verifyIconImage(icon_filename) = 0
 then     --- アイコンファイルに問題が存在する。
111       alert("Error: アイコンファイル" && msgHand
ler_filename "が無効です。既定のアイコンを使用します。")
112       insert_iconFileName(SG_TYPETABLE
, mimeToInstall, DEFAULT_ICONFILE)
113   else
114       insert_iconFileName(SG_TYPETABLE
, mimeToInstall, icon_Filename)
115   endif
116
117   if filetype <> "" then
118       insert_fileytype(mimeToInstall, f
iletype)
119       writeTypeToRegistry(mimeToInstal
l, filetype)
120   end if
121
122   set retVal = write_TypeTable_File()
123   if retVal = 0 then
124   alert("ファイルへのタイプテーブルの書き込みにエラーが発生しました。" && mimeToInstall && "はインストールされていません。")
125   return(0)
126   else return(1)
127
128   end  --- Install_type
```

```
１２９
１３０
１３１　－－－ Ｕｎｉｎｓｔａｌｌ＿Ｔｙｐｅは、ファイルとグローバル変数ＳＧ＿ＴＹＰＥＴＡＢＬＥ
１３２　の両方からＭＩＭＥタイプとそのプロパティを削除する。
１３３
１３４　ｏｎ ｕｎＩｎｓｔａｌｌ＿Ｔｙｐｅ ｍｉｍｅＴｙｐｅ
１３５　ｇｌｏｂａｌ ＳＧ＿ＴＹＰＥＴＡＢＬＥ －－－ インストールされているすべてのＭＩＭＥタイプに関する情報である。
１３６
１３７　ｓｅｔ ＳＧ＿ＴＹＰＥＴＡＢＬＥ ＝ ［：］　－－－ ＭＩＭＥタイプ情報のプロパティリストを初期化する。
１３８
１３９　－－－ 既存のＭＩＭＥタイプ情報をＲＡＭに読み込む。
１４０　ｓｅｔ ＳＧ＿ＴＹＰＥＴＡＢＬＥ ＝ Ｒｅａｄ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅ（ｔｈｅ ｐａｔｈｎａｍｅ ＆ ”ｔｙｐｅｔａｂｌｅ．ｔｘｔ”）
１４１
１４２　ｉｆ ｃｏｕｎｔ（ＳＧ＿ＴＹＰＥＴＡＢＬＥ）＝ ０ ｔｈｅｎ　－－－ ｔｙｐｅｔａｂｌｅファイルの読み取りに失敗した。
１４３　　ａｌｅｒｔ（”エラー：ＭＩＭＥタイプファイルの読み取りに失敗しました。”）
１４４　　ｒｅｔｕｒｎ（０）
１４５　ｅｎｄ ｉｆ
１４６
１４７　ｄｅｌｅｔｅ＿ｍｉｍｅｔｙｐｅ（ｍｉｍｅＴｙｐｅ）
１４８
１４９　－－－ 改訂されたＴｙｐｅｔａｂｌｅをファイルに書き込む。
１５０　ｓｅｔ ｒｅｔＶａｌ ＝ ｗｒｉｔｅ＿ＴｙｐｅＴａｂｌｅ＿Ｆｉｌｅ（）
１５１　ｉｆ ｒｅｔＶａｌ ＝ ０ ｔｈｅｎ
１５２　ａｌｅｒｔ（”エラー：” ＆＆ ｍｉｍｅＴｏＩｎｓｔａｌｌ ＆＆ ”をアンインストールできませんでした。
１５３　Ｔｙｐｅｔａｂｌｅファイルの書き込みエラーです”）
１５４　ｒｅｔｕｒｎ（０）
１５５　ｅｌｓｅ ｒｅｔｕｒｎ（１）
１５６
１５７　ｅｎｄ ｕｎＩｎｓｔａｌｌ＿Ｔｙｐｅ
１５８
１５９　－－－ Ｗｒｉｔｅ＿Ｔｙｐｅｔａｂｌｅ＿Ｆｉｌｅは、ＳＧ＿ＴＹＰＥＴＡＢＬＥ内の情報をディスク
１６０　上のＴｙｐｅｔａｂｌｅファイルに書き込む。このファイルは、各ＭＩＭＥタイプに関連するプロパティを保存している。
１６１　－－－ ＳＧ＿ＴＹＰＥＴＡＢＬＥは、ＭＩＭＥタイプのリストを格納するプロパティリストである。
１６２
１６３　ｏｎ Ｗｒｉｔｅ＿Ｔｙｐｅｔａｂｌｅ＿Ｆｉｌｅ
１６４　ｇｌｏｂａｌ ＳＧ＿ＴＹＰＥＴＡＢＬＥ －－－ 様々なＭＩＡＷで共有するスーパーグローバル変数である。
１６５　ｓｅｔ ｆｉｌｅＮａｍｅ ＝ ｔｈｅ ｐａｔｈｎａｍｅ ＆ ”ｔｙｐｅｔａｂｌｅ．ｔｘｔ”
１６６　ｓｅｔ ｂｋｕｐＦｉｌｅＮａｍｅ ＝ ｔｈｅ ｐａｔｈｎａｍｅ ＆ ”ｔｙｐ
```

```
etable.bak"
167
168  if count(SG_TYPETABLE) = 0 then  --- MIMEタイプが定義されていない。
169      alert("エラー：書き込むMIMEデータが存在していません。")
170  return(0)
171  end if
172
173  --- typetableファイルのバックアップを作成する。
174  copyFile(filename, bkFileName)
175
176  --- Fileio Xtraを起動する。
177      set mFile = new(xtra "fileio")
178
179      set retVal = deleteFile(mFile, filename)  --- 書き換えの前に古いバージョンを削除する。
180      set retVal = createFile(mFile, Filename)
181      if retVal = 0 then
182          alert("typetableファイルの更新にエラーが発生しました。")
183          renameFile(bkFileName, filename)
184          return(0)
185      end if
186
187      openFile(mFile, fileName, 2)  --- 書き込みアクセス用に開く。
188      setPosition(mFile, 0)
189
190  --- データをファイルに書き込む。
191  set i = 1
192  set mimeType = getAt(SG_TYPETABLE, i)
193  repeat while mimeType <> 0
194      set dataToWrite = mimetype
195      put " " & get_filetype(mimetype) into dataToWrite
196      put " " & get_iconFileName(mimetype) into dataToWrite
197  put " " & get_MsgHandler(mimetype) into dataToWrite
198  writeline(mFile, dataToWrite)
199  set i = i + 1
200  set mimeType = getAt(SG_TYPETABLE, i)
201  end repeat
202
203  closeFile(mFile)
204  set retVal = deleteFile(mFile, bkFileName)  --- バックアプファイルを削除する。
```

```
205    return(0)
206    end Write_TypeTable_File
207    --- Read_Typetable_Fileはtypetableファイルを読み取り、
208    --- メモリ内にデータ構造SG_TYPETABLEを作成する。
209    --- SG_TYPETABLEは、MIMEタイプのリストを格納するプロパティリストである。
210
211    この関数から戻ると、グローバルプロパティリストデータ構造SG_TYPETABLEは、
212    各MIMEタイプごとに1つのエントリを格納しており、メッセージ処理ムービー用のpathNameとIconFileが含まれている。
213    後で、このデータ構造にRAM内のアイコンのキャストメンバ番号を追加する。
214    現時点では、これらはすべて0に設定されている。
215    このデータ構造は、["text/plain":["txt", 0, "C:¥KidCode¥text.gif", "C:¥KidCode¥text.dxr"],
216    "x-application/grid":["", 0, "C:¥KidCode¥grid.gif", "C:¥KidCode¥grid.dxr"]]のようになっている。
217
218    on Read_TypeTable_File
219    global SG_TYPETABLE --- 様々なMIAWで共有するスーパーグローバル変数である。
220    set fileName = the pathname & "typetable.txt"
221
222    set SG_TYPETABLE = [:] --- MIMEタイプのプロパティリストを初期化する。
223
224    --- Fileio Xtraを起動する。
225    set mFile = new(xtra "fileio")
226    openFile(mFile, fileName, 1) --- 読取専用アクセス用に開く。
227    set status = status(mFile)
228
229    if status <> 0 then
230        alert("エラー:MIMEタイプテーブルが開けませんでした。"& error(mFile, status))
231        closeFile(mFile) --- 安全のため。
232        return FALSE
233    end if
234
235    setPosition(mFile, 0)
236
237    --- LINGOでは行単位の読み取りができない。従って、ファイル全体を
238    文字列strに読み込むことにより、これをシミュレートしている。
239        set str = readFile(mFile)
240
```

```
241       set nTypes = the number of lines in
str
242           repeat with j = 1 to nTypes
243               set mimetype = word 1 of line
j
244               insert_mimetype(mimetype)
245               insert_filetype(mimetype, w
ord 2 of line j)
246               insert_iconFileName(mimety
pe, word 3 of line j)
247               insert_msgHandler(mimetype
, word 4 of line j)
248           end repeat
249
250       closeFile(mFile)
251
252   end Read_TypeTable_File
253
254
255   on Read_Icon_Files_To_RAM
256   global SG_TYPETABLE
257   global SG_DEFAULT_ICON_PTR = 1000
258
259   if count(SG_TYPETABLE) = 0 then --- MI
MEタイプが定義されていない。
260       alert("エラー：MIMEタイプデータが存在していません。ア
イコンをロードできません。")
261   return(0)
262   end if
263
264   -- 既定のアイコンをロードする。
265   importFileInto(member SG_DEFAULT_ICO
N_PTR, the pathname & "defaulticon.gif")
266
267   --- MIMEタイプを一巡してアイコンをロードする。
268   set castNum = SG_DAFAULT_ICON_PTR + 1
--- 既定のものの直後の
269   アイコンである。
270   set i = 1
271   set mimeType = getAt(SG_TYPETABLE, i)
272   repeat while mimeType <> 0
273       set iconFile = get_iconFileName(SG
_TYPETABLE, mimetype)
274       if icon = "" then --- アイコンが定義されておらず
、既定のものを使用する。
275           set iconPTR = SG_DEFAULT_ICON_P
TR
276       else
277   set iconPTR = read_IconFile(iconFile,
 castNum)
```

```
278  if iconPtr > 0 then
279      set castNum = castNum + 1
280  else set iconPtr = SG_DEFAULT_ICON_PTR
281      end if
282
283      insert_iconPtr(mimetype, iconPtr)
284  set i = i + 1
285  set mimeType = getAt(SG_TYPETABLE, i)
286  end repeat
287
288  return(1)
289
290  end Read_Icon_Files_To_RAM
291
292
293
294  --- Read_IconFileは、1つのアイコンビットマップをRAMにロードする。
295  on read_IconFile filename, castMemberNum
296  set retVal = importFileInto(member castNum, iconFile)
297  return(retVal)
298  end read_IconFile
299  --- MIMEタイプ情報のデータアクセス関数
300  --- データは、「MIMEタイプ：プロパティ」という構造で
301  --- プロパティリストSG_TYPETABLE内に保存される。
302  --- ここで、MIMEタイプは文字列である（例：”text／plain”）。
303  --- プロパティは、次の要素を有するリストである。
304  --- [filetype, iconPtr, idonFilename, msgHandler_FileName]
305  --- 例：SG_TYPETABLE =
306  --- [”text／plain”:[”txt”, 0, ”C:¥KidCode¥text.gif”, ”C:¥KidCode¥text.dxr”],
307  --- ”x-application／grid”:[””, 0, ”C:¥KidCode¥grid.gif”, ”C:¥KidCode¥grid.dxr”]]
308
309  on get_mimetype mimetype
310  global SG_TYPETABLE
311  return(getProp(SG_TYPETABLE, mimetype))
312  end
313
314  on get_filetype mimetype
315  global SG_TYPETABLE
316  set theProperties = getProp(SG_TYPETABLE, mimetype)
```

```
317   return(getAt(theProperties, 1))
318   end
319
320   on get_iconPtr mimetype
321   global SG_TYPETABLE
322   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
323   return(getAt(theProperties, 2))
324   end
325
326   on get_iconFileName mimetype
327   global SG_TYPETABLE
328   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
329   return(getAt(theProperties, 3))
330   end
331
332   on get_msgHandler mimetype
333   global SG_TYPETABLE
334   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
335   return(getAt(theProperties, 4))
336   end
337
338   on insert_mimetype mimetype
339   global SG_TYPETABLE
340   addProp(SG_TYPETABLE, mimetype)
341   end
342
343   on insert_filetype mimetype, filetype
344   global SG_TYPETABLE
345   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
346       add(theProperties, filetype)
347       setProp(SG_TYPETABLE, mimetype, t
heProperties)
348   end
349
350   on insert_iconPtr mimetype, iconPtr
351   global SG_TYPETABLE
352   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
353       add(theProperties, iconPtr)
354       setProp(SG_TYPETABLE, mimetype, t
heProperties)
355   end
356
357   on insert_iconFileName mimetype, icon
Filename
358   global SG_TYPETABLE
```

```
359   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
360       add(theProperties, iconFilename)
361       setProp(SG_TYPETABLE, mimetype, t
heProperties)
362   end
363
364   on insert_msgHandler mimetype
365   global SG_TYPETABLE
366   set theProperties = getProp(SG_TYPETA
BLE, mimetype)
367       add(theProperties, msgHandler)
368       setProp(SG_TYPETABLE, mimetype, t
heProperties)
369   end
370
371
372   on delete_mimetype mimetype
373   global SG_TYPETABLE
374   deleteProp(SG_TYPETABLE, mimetype)
375   end
376
377   on delete_filetype mimetype
378   global SG_TYPETABLE
379   set properties = getProp(SG_TYPETABLE
, mimetype)
380       setAt(properties, 1, "")
381       setProp(SG_TYPETABLE, mimetype, p
roperties)
382   end
383
384   on delete_icon mimetype
385   global SG_TYPETABLE
386   set properties = getProp(SG_TYPETABLE
, mimetype)
387       setAt(properties, 2, 0)
388       setProp(SG_TYPETABLE, mimetype, p
roperties)
389   end
390
391   on delete_iconFileName mimetype
392   global SG_TYPETABLE
393   set properties = getProp(SG_TYPETABLE
, mimetype)
394       setAt(properties, 3, "")
395       setProp(SG_TYPETABLE, mimetype, p
roperties)
396   end
397
398   on delete_msgHandler mimetype
```

```
399  global SG_TYPETABLE
400  set properties = getProp(SG_TYPETABLE, mimetype)
401      setAt(properties, 4, "")
402      setProp(SG_TYPETABLE, mimetype, properties)
403  end
```

【図面の簡単な説明】

【図１】従来技術の電子メールボックスディスプレイヤの画面ショットである。

【図２】本発明による電子メールボックスディスプレイヤの画面ショットである。

【図２ａ】本発明の別の実施例による電子メールボックスディスプレイヤの画面ショットである。

【図３】本発明による電子メールクライアントのコンポーネント構成のブロックダイアグラムである。

【図４】本発明によるメールボックスディスプレイヤの基本的な動作を示す概略フローチャートである。

【図５】本発明によるメールボックスディスプレイヤの別の実施例の基本的な動作を示す概略フローチャートである。

【図６】メールアイテムの様々なプロパティ用のアイコンのルックアップを示す概略フローチャートである。

【図６ａ】ＭＩＭＥタイプ用のアイコンのルックアップを示す概略フローチャートである。

【図６ｂ】添付ファイルのファイルタイプ用のアイコンのルックアップを示す概略フローチャートである。

【図７】タイプアップデータコンポーネントの一実施例におけるｉｎｉｔｉａｌｉｚｅ＿ＴｙｐｅＴａｂｌｅ関数を示す概略フローチャートである。

【図８】タイプアップデータコンポーネントの一実施例におけるｒｅａｄ＿ＴｙｐｅＴａｂｌｅ＿ｆｉｌｅ関数を示す概略フローチャートである。

【図９】タイプアップデータコンポーネントの一実施例におけるｗｒｉｔｅ＿ＴｙｐｅＴａｂｌｅ＿ｆｉｌｅ関数を示す概略フローチャートである。

【図１０】タイプアップデータコンポーネントの一実施例におけるｒｅａｄ＿ｉｃｏｎ＿ｆｉｌｅｓ＿ｔｏ＿ＲＡＭ関数を示す概略フローチャートである。

【図１１】タイプアップデータコンポーネントの一実施例におけるｉｎｓｔａｌｌ＿ＴＹＰＥ関数を示す概略フローチャートである。

【図１】

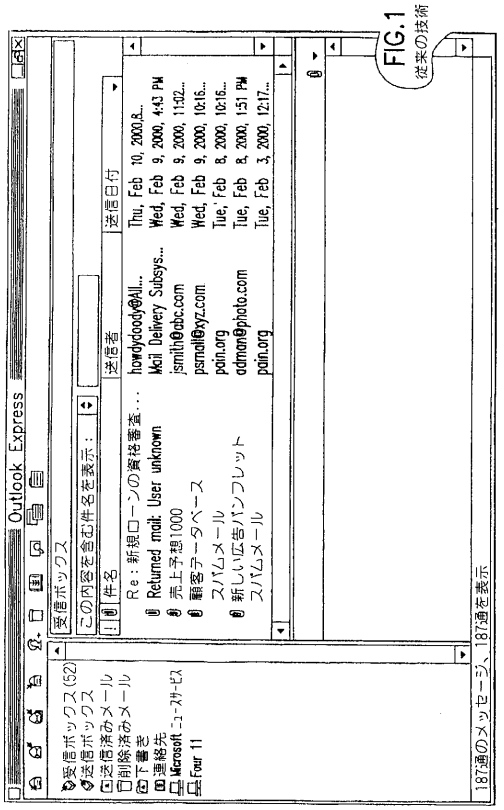


FIG.1
従来の技術

【図２】

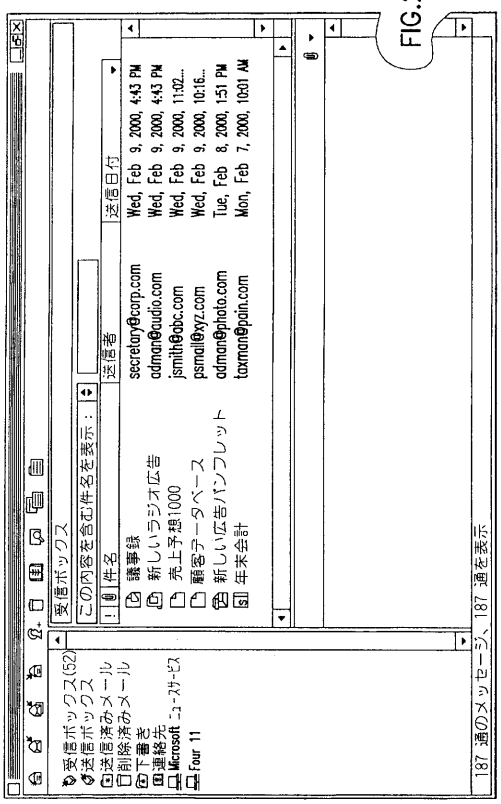


FIG.2

【図２ａ】

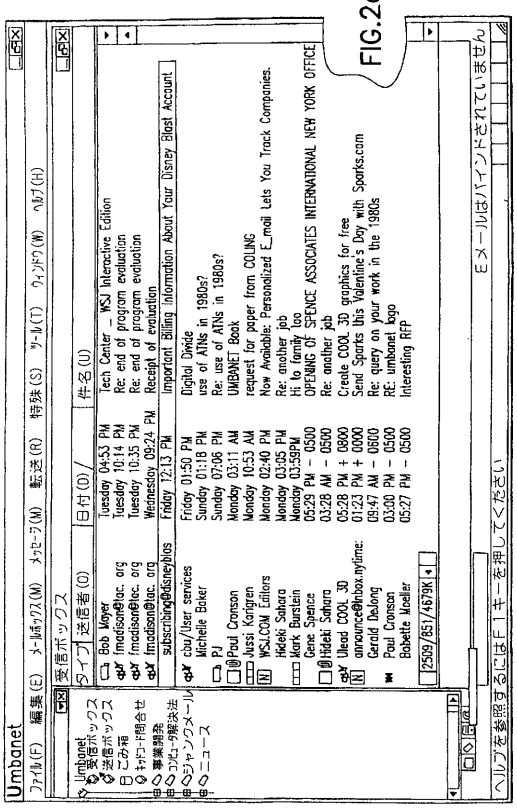


FIG.2a

【図３】

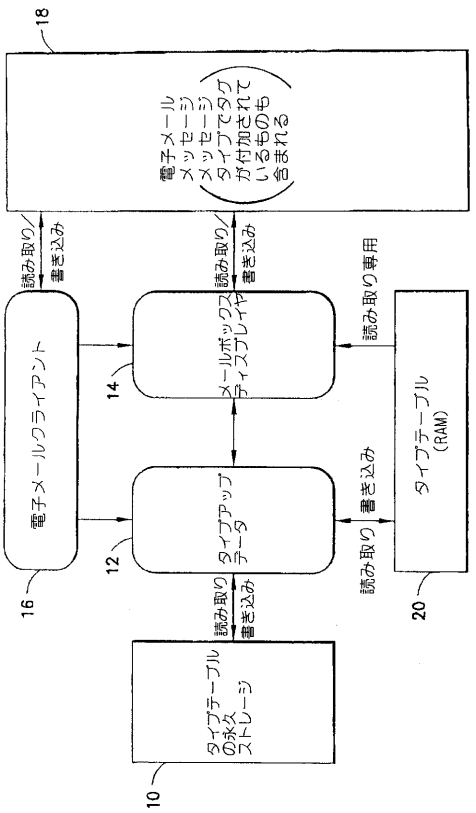


FIG.3

【図４】

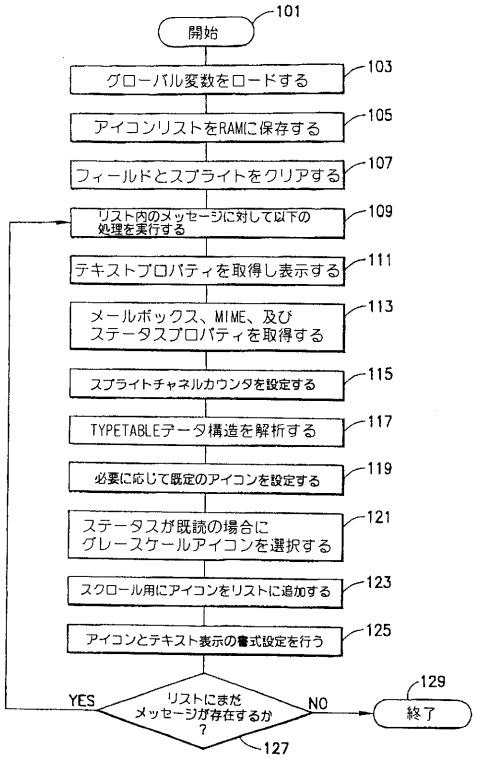


FIG.4

【図５】

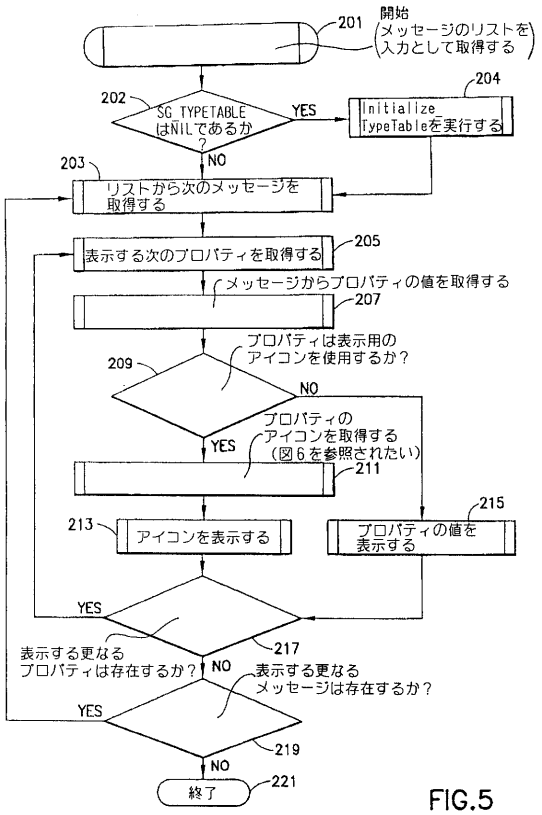


FIG.5

【図６】

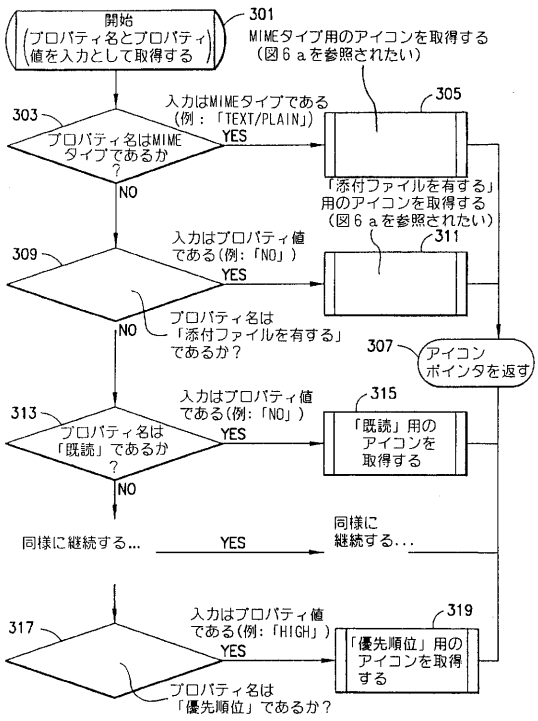


FIG.6

【図６ａ－１】

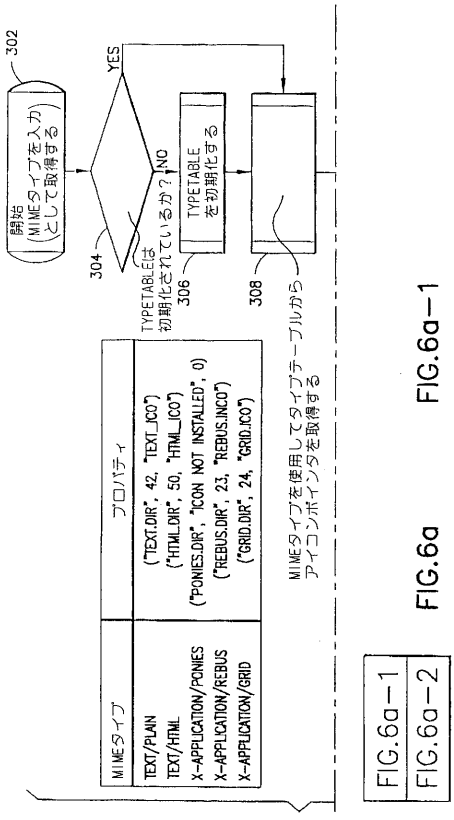


FIG.6a-1

FIG.6a

FIG.6a－1
FIG.6a－2

【図６ａ－２】

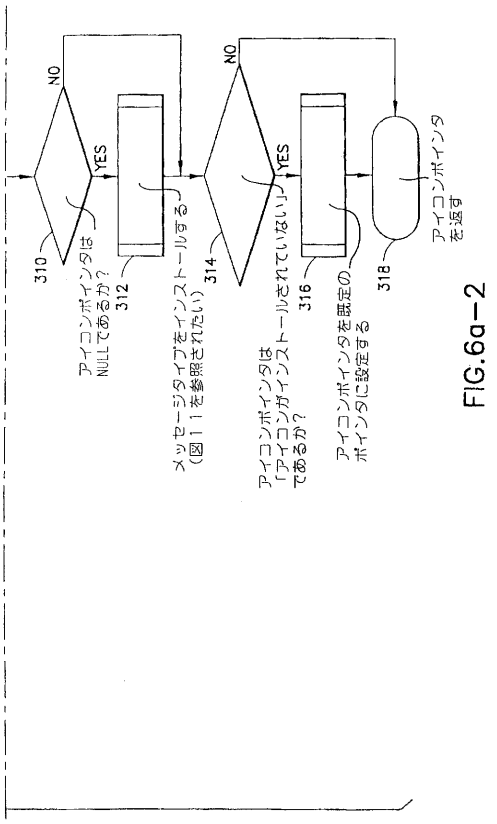


FIG.6a-2

【図６ｂ－１】

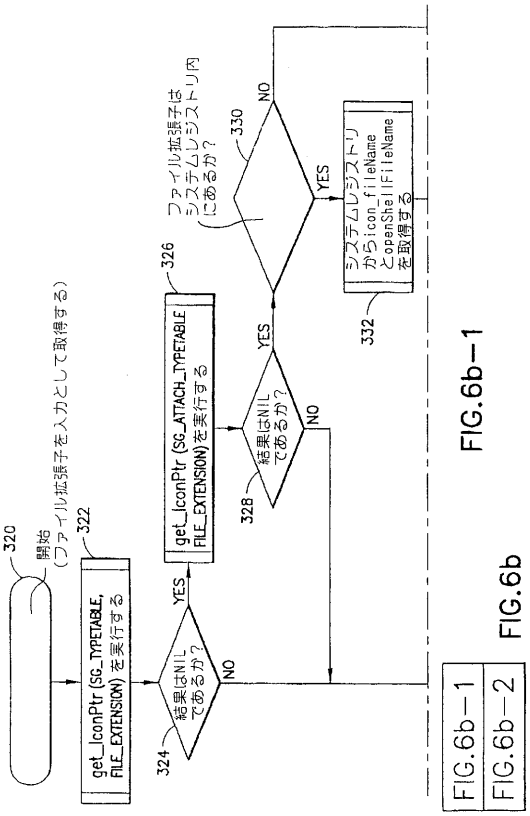


FIG.6b-1

FIG.6b

FIG.6b-1
FIG.6b-2

【図６ｂ－２】

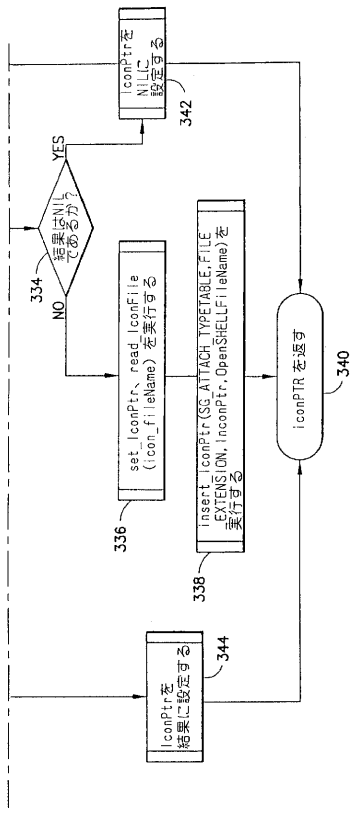


FIG.6b-2

【図７】

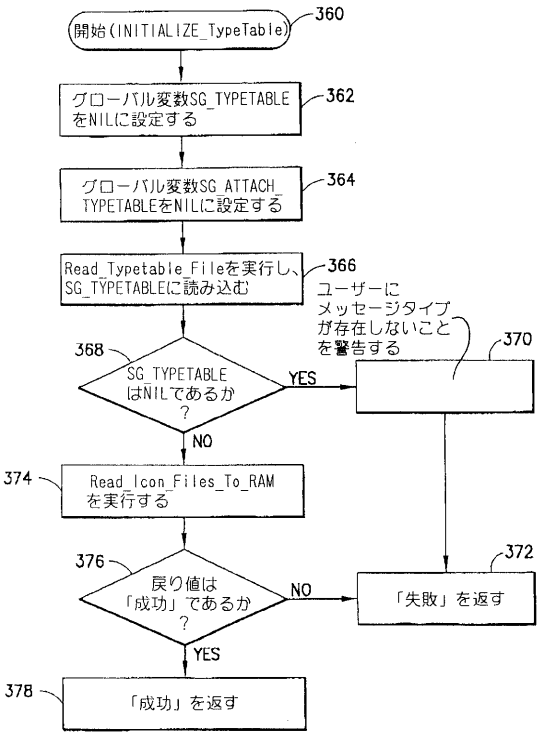


FIG.7

【図８】

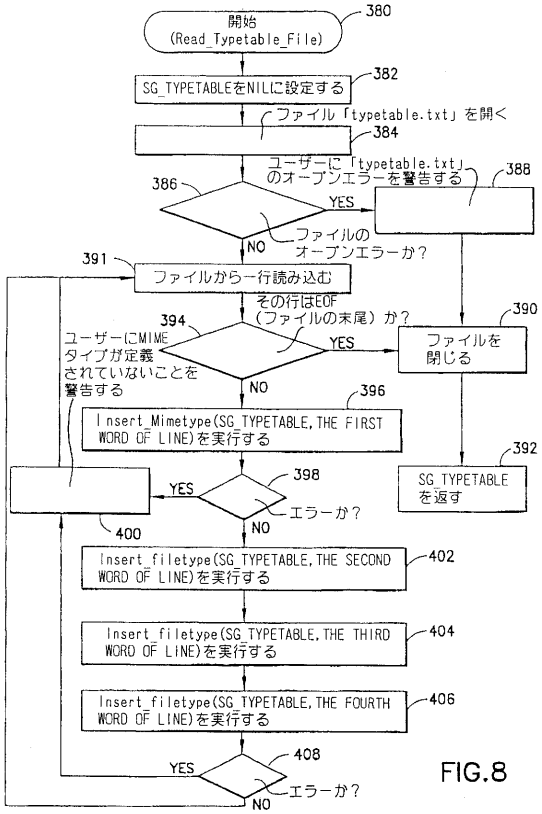


FIG.8

【図９Ａ】

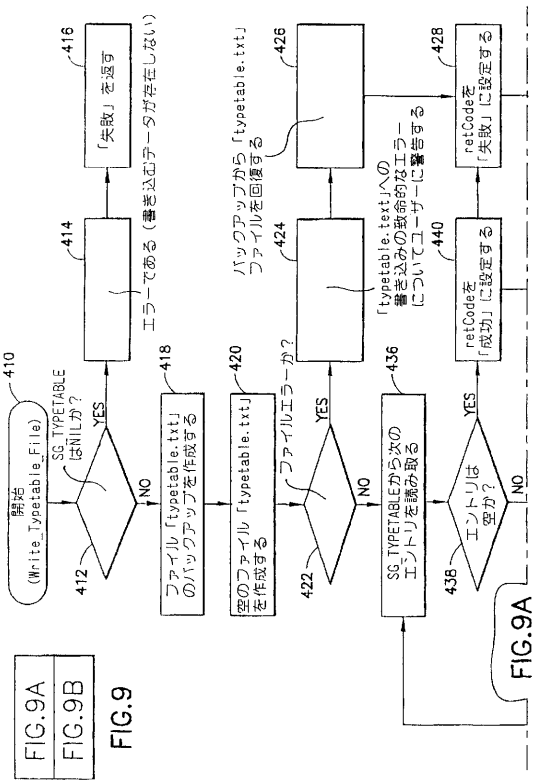


FIG.9A

FIG.9

FIG.9A
FIG.9B

【図９Ｂ】

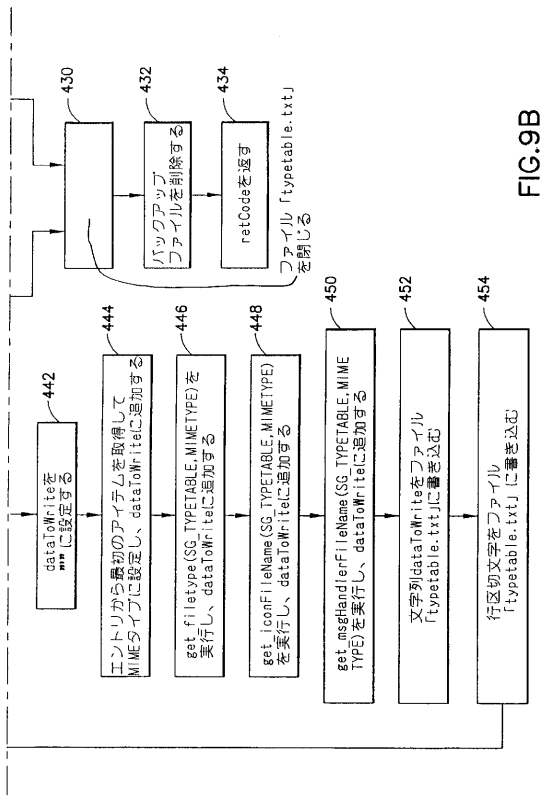


FIG.9B

【図１０Ａ】

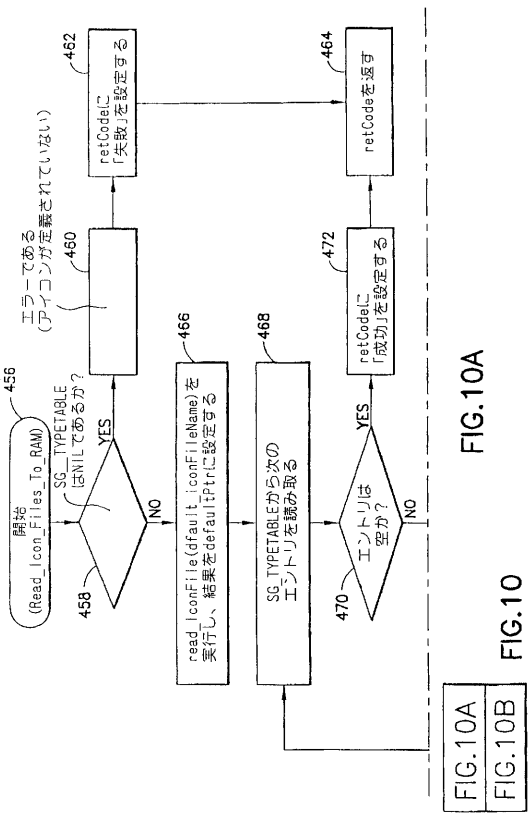


FIG.10A

FIG.10

FIG.10A
FIG.10B

【図１０Ｂ】

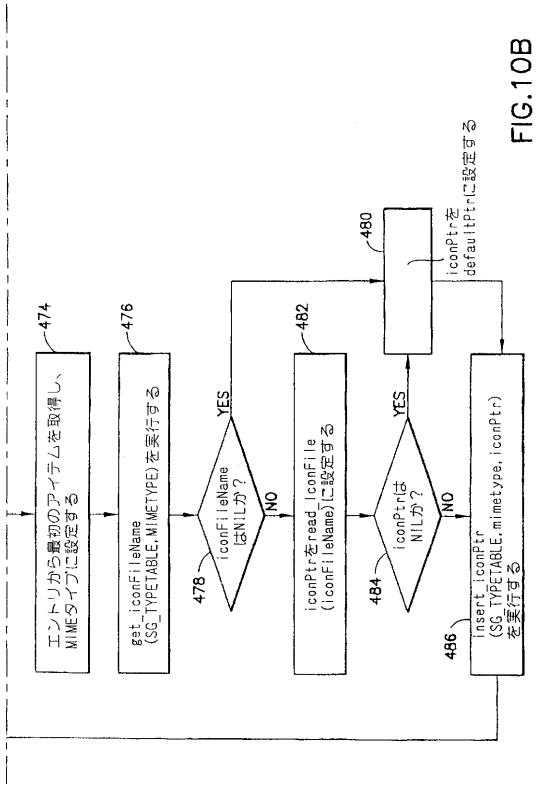


FIG.10B

【図１１Ａ】

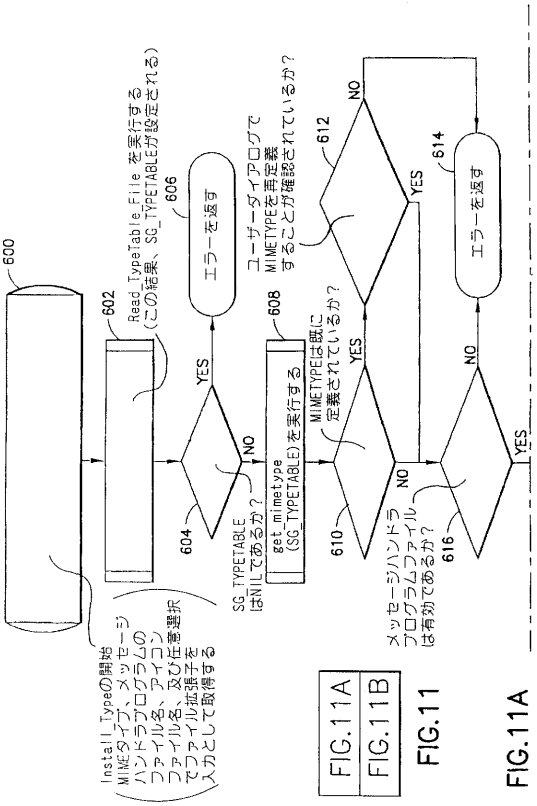


FIG.11A

【図１１Ｂ】

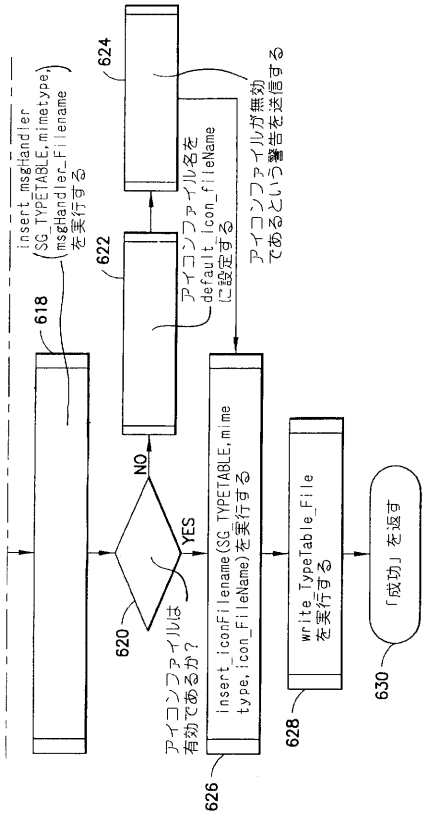


FIG.11B

FIG.11A
FIG.11B

FIG.11

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 January 2002 (03.01.2002)

PCT

(10) International Publication Number
WO 02/01373 A1

(51) International Patent Classification⁷: G06F 13/00, 15/16

(21) International Application Number: PCT/US01/20348

(22) International Filing Date: 20 June 2001 (20.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/604,426 27 June 2000 (27.06.2000) US

(71) Applicant: **INTELLINET, INC.** [US/US]; 325 Riverside Drive, New York, NY 10025 (US).

(72) Inventor: **BAKER, Michelle**; 325 Riverside Drive, New York, NY 10025 (US).

(74) Agents: **GORDON, David, P.** et al.; 65 Woods End Road, Stamford, CT 06905 (US).

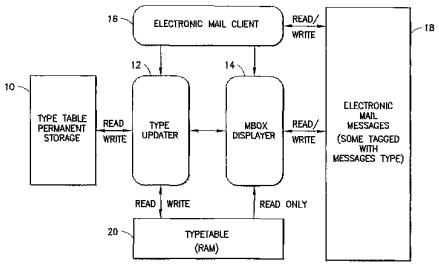
(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CO, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** ENHANCED ELECTRONIC MAIL SYSTEM INCLUDING METHODS AND APPARATUS FOR IDENTIFYING MIME TYPES AND FOR DISPLAYING DIFFERENT ICONS



(57) **Abstract:** Electronic mail client software (16) has a mailbox displayer (14) which lists messages together with an icon for each message where the icon is associated with the MIME type of the message. Mail which contains file attachment (18) is listed in the inbox with an icon indicative of the type of file attached to the mail. The mailbox displayer interprets the MIME type and selects the appropriate icon either from the icon registry in the OS or from a directory of icons maintained by the email client software. If there is no appropriate icon in the directory of icons, the mailbox displayer uses icon image data contained in a subpart of the MIME message if it is available. Otherwise, no icon or a generic icon is used. According to the presently preferred embodiment, a type table (20) is maintained by a type updater (12) component. The type includes a list of message types and subtypes together with filenames of scalable icons to be used by the mailbox displayer.

WO 02/01373 A1

ENHANCED ELECTRONIC MAIL SYSTEM INCLUDING METHODS AND APPARATUS FOR IDENTIFYING MIME TYPES AND FOR DISPLAYING DIFFERENT ICONS

This application is a continuation-in-part of application serial number 09/209,162 filed December 10, 1998, the complete disclosure of which is hereby incorporated by reference herein. This application is related to copending application serial number [BAK-007] filed simultaneously herewith, the complete disclosure of which is also hereby incorporated by reference herein.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to an electronic mail program. More particularly, the invention relates to an electronic mail program having a mailbox browser display which displays different icons for different types of mail item MIME types.

2. State of the Art

In recent years electronic mail ("email") has become widely used in business, education, and in personal communications. One of the features of electronic mail which is most convenient, particularly in business and in education, is the ability to attach a binary computer file to an email message. This feature enables email correspondents to rapidly share word processing documents, database documents, spreadsheet documents, multimedia documents, or virtually any kind of binary file created by a computer. There are, however, some serious limitations and inconveniences associated with attaching a binary file to an email message.

The original Internet mail system as defined in 1982 with RFC (Request for Comments) 821 and 822 had a number of important limitations. In particular, the system was not designed to carry large quantities of arbitrary data in an email message. In fact, the 1982 SMTP (Simple Mail Transport Protocol) standard required that an email message consist of a single message containing only ASCII characters in lines of 1000 characters (blocks of 32k) or less.

The ability to send binary data through the Internet electronic mail system was made possible with the MIME (Multipurpose Internet Mail Extensions) standard for Internet messages. The original MIME standard was published as an Internet Request For Comments document (RFC 1341) and approved in June of 1992. (See Internet RFCs 2045,2046, and 2047

for the latest MIME standards documents.) The MIME standard describes how an email message should be formatted in order to be considered MIME compliant. MIME defines a set of message header fields and a set of message encoding standards that are designed to overcome the limitations of RFC 822 message formats and still be transportable through any of the numerous legacy mail transport systems in use on the Internet. (See specifically, N. Freed and N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part 1: Format of Message Bodies, Network Working Group, Request For Comments (RFC 2045) November 1996.) MIME message header fields extend those defined in RFC 822 and describe the content and encoding type of the email message. Encoding schemes allowed in the MIME standard include "quoted-printable", and "base64". In addition, three unencoded data types are allowed. These are labeled "8bit", "7bit", or "binary". It should be noted that legacy gateways still do not handle binary data and nearly all MIME compliant messages encode binary data as "7bit", the default encoding for MIME.

Today MIME is implemented in all of the major electronic mail clients or "User Agents", e.g. Microsoft Outlook and Outlook Express, Netscape Communicator, and Qualcomm Eudora. However, only a few MIME types including "text/plain", "text/html", "multipart/alternative", and "multipart/mixed" can be handled by these programs. Probably the most important feature of the MIME standard that was that it allowed any binary data to be appropriately encoded and sent through the older SMTP system of mail gateways and exchanges. Mail client programs such as those listed above were modified to allow users to attach any type of file to a mail message. This was done by (a) including an appropriate encoding module to translate the binary data of an arbitrary file to an acceptable MIME encoding such as "7bit" or "base64", (b) expanding the Mail client's ability to handle messages with a MIME type set to "multipart", and (c) including the file specified by a user as a part of the "multipart" message. For many years, mail client programs offered users only the two choices; they could send a simple text message (sent with "content-type = text/plain") or they could attach any file to a simple text message (sent with "content-type = multipart/mixed").

More recently the programs listed above have been extended to allow authors to use basic types of text formatting such as alternative fonts and styles by including these features in the mail client text editor and sending the message with a MIME type set to "text/html". Today Microsoft's Outlook even allows a person to use Word, a full featured text editor, to author electronic mail messages by converting the Word file format to HTML before manually inserting it into the body of the mail message for sending. Nevertheless, mail client programs

still rely exclusively on file attachments with message MIME types set to "multipart" for any other type of file format.

If the sender and the receiver of the email message with the attached binary file are using the same brand and version of email program and both programs are configured in substantially the same way, the receiver's email program should automatically apply the appropriate decoding to the attached binary file and produce a file which is identical to the file which was attached to the email by the sender. However, if the sender and receiver are using different email programs, the recipient may receive a file which must be decoded by the recipient using a separate decoding program.

Even after the file is properly received and decoded, it is often difficult for the receiver of the file to open the file. The receiver of the file might expect that "clicking" on the file icon will open the file. However, clicking on the file icon will often not open the file. It may result in an error message like "application not found" or, worse, it may result in the file being opened by an inappropriate application thereby displaying "gibberish". The receiver of the file must have a program capable of reading (opening) the file. For example, if one attaches a spreadsheet file to an email message, the receiver of the file must have a spreadsheet program in order to open the file. Technically, it is not necessary that the receiver of the file have the same brand program as that which created the file. However, opening a file with a program which did not create it, though possible, can be very inconvenient. The receiver of the file must know what kind of file is attached to the email message, must know what program on their computer is capable of reading that type of file, must launch the program, must open the file from within the program, and wait while the program translates the file.

The limitations of Internet electronic mail can become even more frustrating if the sender and recipient are not using the same operating system (OS). Some mail attachment encoding schemes (and file compression schemes) are OS-dependent and it is possible that an email recipient could receive a file which is impossible to decode (or decompress).

These limitations in electronic mail have discouraged many people, particularly non-sophisticated computer users, from attaching files to electronic mail messages. In fact, for some novice users, the task of launching one application to create a document, saving the document, launching a separate email application to create an email message, and then locating the saved document for attachment to an email message is daunting enough to discourage them. In

addition, novice users often complain that after "downloading" a file attached to an email message they cannot find the file on their hard disk.

Most email client software allows the user to sort items in the inbox by sender, subject, or date in order to locate more easily a particular mail item. In addition, most email client software indicates whether a particular message includes an attached file. This is indicated by an icon such as a paper clip icon or a generic document icon or a floppy disk icon, for example. However, the same icon is used regardless of the nature of the attachment and there is no way of knowing the nature of the attachment until the message is opened. Prior art Figure 1 shows an example of a typical email inbox where some of the mail items have attached files indicated by the paper clip icon to the left of the subject name. Though not specifically shown in Figure 1, those skilled in the art will appreciate that generic icons, such as !, a, Â , 4, etc., may also be displayed alongside the message subject to indicate various "properties" of the message, such as whether it is a high priority message, whether you have already replied to the message, etc. These generic icons are usually monochromatic font characters taken from a "dingbats" font or the like.

In the most recent versions of the major email client programs, an icon that represents the file type of an attached file is displayed in the body of the mail message after the message is opened by the user. This is possible because computer operating systems such as Microsoft Windows or Macintosh OS maintain data that associates information with each file type known to the system. This information includes a graphical icon and the location of programs that may be used to "open", "edit", or to perform a handful of other actions on the file. For example, in Microsoft Windows the system registry includes entries for each file type that is known to the system and at least some of the information described above is associated with the file type. When a user opens an electronic mail message with "content-type = multipart/mixed", a mail client program built for Microsoft Windows (e.g. Microsoft Outlook) determines that the second part of the message was an attached file, identifies a line of text within the message such as, Attachment Converted: "c:\attach\aFile.doc", looks in the system registry for the icon associated with the file type ".doc", and displays the graphical icon inside the body of the message.

In current systems, MIME type is not used to associate icons to files, rather the file type extension is used. This creates important limitations in the ability to associate different versions of software or documents created by different versions of the software with different icons. For example all documents created by MS Word, regardless of which version of Word was used,

have the same file type (file extension) and as a result are associated with the same icon. This is true even though many newer versions of the files cannot be read by older versions of the software.

My previously incorporated parent application discloses electronic mail software which includes a main email component and a number of installable components. The installable components include authoring/reading components for creating/reading different kinds of documents and mailbox components for listing different kinds of messages or for listing messages in different styles. The main email component provides an underlying graphical user interface for functions directly associated with the storage and transfer of electronic mail messages, and also handles all data bundling and unbundling required to transform a message created by an authoring component into a MIME compliant message. The authoring/reading components act like applications embedded within the email program and allow specific types of documents such as spreadsheets, graphics, databases, etc. to be created from within the email program and emailed directly. The authoring/reading components also allow received documents to be read without the difficulties traditionally associated with attaching binary files to an email letter. The authoring components of the invention pass data to the main email component which packages the data as a MIME compliant message. When the message is received, the main email component concatenates (as needed) and decodes the MIME message and sends the data to the authoring/reading component associated with the MIME type.

My previously incorporated parent application broadly disclosed and claimed mailbox handling software whereby messages of different types are displayed in different ways in a mailbox listing within the context of the modular component email software.

It is believed that certain features disclosed in my parent application are applicable to any email client software and may be used to improve the process of attaching files to email and using files attached to email.

SUMMARY OF THE INVENTION

It is therefore an object of the invention to provide an electronic mail program which includes an inbox list whereby different kinds of messages and attached documents are displayed with different kinds of icons.

In accord with this object which will be discussed in detail below, electronic mail client software according to the invention has a mailbox displayer which lists messages together with an icon for each message where the icon is associated with the MIME type of the message. Mail which contains a file attachment is listed in the inbox with an icon indicative of the type of file attached to the email. The mailbox displayer interprets the MIME type and selects the appropriate icon either from the icon registry in the OS or from a directory of icons maintained by the email client software. For example, if an email with an ADOBE ACROBAT file attachment is received, the ADOBE ACROBAT icon will appear in the mailbox listing alongside the mail item listing. In addition, if a message is created with a special authoring/reading component as described in my parent application, the icon associated with the authoring/reading component will be displayed in the mailbox listing as part of the line displaying the mail item.

The electronic mail software of the present invention is described by example with reference to the email software of my parent application which includes a main email component and a number of installable components which communicate bidirectionally with the email component through an application programming interface (API). The installable components include authoring/reading components as well as a mailbox displayer component. According to the presently preferred embodiment, a component is also included for maintaining a database of icons.

The mailbox displayer component functionality is invoked by the user when the mailbox is opened, when the list of mail is scrolled, etc. The mailbox displayer component preferably includes all of the functionality of state-of-the-art mailbox displayers and includes the functionality of looking to a directory of icons for display with information about the message based on the MIME type of the message. In the Lingo embodiment, a data structure is created for each message with an additional TYPE field that is based on the MIME type and subtype of the message. The internal TYPE field is used to associate MIME types to icons. Another embodiment uses the contents of "content-type" (MIME type) header of the message directly to associate with icon images. If there is no appropriate icon in the directory of icons, the mailbox displayer uses icon image data contained in a subpart of the MIME message if it is available. Otherwise, no icon or a generic icon is used. According to the presently preferred embodiment, a type table is maintained by a type updater component. The type table includes a list of message types and subtypes together with filenames of scalable icons to be used by the mailbox displayer. The invention prefers scalable icons so that the icon can be sized to accompany the font size chosen to display the mailbox contents.

Several embodiments of the type updater component are provided. According to the first embodiment, icons are installed/removed manually by the user. According to a second embodiment, icons are automatically installed/removed when modular authoring/reading components are installed/removed. According to a third embodiment, new icons are added automatically whenever a new message type is encountered by the mailbox displayer. The new icon is retrieved from either the operating system registry or from the icon image data embedded in the message. According to a fourth embodiment, the type updater automatically queries a network server for new icon information and downloads icon image data as needed.

Additional objects and advantages of the invention will become apparent to those skilled in the art upon reference to the detailed description taken in conjunction with the provided figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a screen shot of a prior art electronic mailbox displayer;

Figure 2 is a screen shot of an electronic mailbox displayer according to the invention;

Figure 2a is a screen shot of an electronic mailbox displayer according to an alternate embodiment of the invention;

Figure 3 is a block diagram of the component organization of an electronic mail client according to the invention;

Figure 4 is a simplified flow chart illustrating the basic operation of a mailbox displayer according to the invention;

Figure 5 is a simplified flow chart illustrating the basic operation of an alternate embodiment of the mailbox displayer according to the invention;

Figure 6 is a simplified flow chart illustrating icon lookup for different mail item properties;

Figure 6a is a simplified flow chart illustrating icon lookup for MIME type;

Figure 6b is a simplified flow chart illustrating icon lookup for filetype of attachments;

Figure 7 is a simplified flowchart illustrating the initialize_TypeTable function of one embodiment of the type updater component;

Figure 8 is a simplified flowchart illustrating the read_TypeTable_file function of one embodiment of the type updater component;

Figure 9 is a simplified flowchart illustrating the write_TypeTable_file function of one embodiment of the type updater component;

Figure 10 is a simplified flowchart illustrating the read_icon_files_to_RAM function of one embodiment of the type updater component; and

Figure 11 is a simplified flowchart illustrating the install_Type function of one embodiment of the type updater component.

BRIEF DESCRIPTION OF THE APPENDICES

Appendix A is a program listing for MACROMEDIA DIRECTOR of a mailbox displayer according to the invention; and

Appendix B is a program listing for MACROMEDIA DIRECTOR of a type updater according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to Figure 2, electronic mail client software according to the invention has a mailbox displayer which lists messages together with an icon for each message where the icon is associated with the MIME type of the message. Mail which contains a file attachment is listed in the inbox with an icon indicative of the type of file attached to the email. For example, as shown in Figure 2, the mail message entitled "Minutes of Meeting" is listed with a MICROSOFT WORD icon indicating that the email message has a WORD file attached to it. The message entitled "New Radio Ad" is listed with a QUICKTIME WAV icon indicating that an audio file is attached to the email. The message entitled "New Ad Brochure" has an attached ADOBE ACROBAT file as indicated by the ACROBAT icon. Similarly, the message

"Sales Forecast" is displayed with an EXCEL spreadsheet icon; the "Customer Database" message is displayed with a FILEMAKER PRO database icon; and the message "Year End Accounting" is displayed with a QUICKEN icon. According to the invention, the mailbox displayer interprets the MIME type of the message and/or the MIME type or document type of the attachment, if any, and selects the appropriate icon either from the icon registry in the OS or from a directory of icons maintained by the email client software.

The electronic mail software of the present invention is described by example with reference to the email software of my previously incorporated parent application which includes a main email component and a number of installable components which communicate bidirectionally with the email component through an application programming interface (API). The installable components include authoring/reading components as well as at least one mailbox displayer component. Figure 2a illustrates an embodiment of the email software of my previously incorporated parent application which displays an icon indicative of the authoring/reading component associated with the mail message. This embodiment is also capable of receiving email from (and sending email to) prior art email clients. Figure 2a illustrates a mailbox list where two mail items are shown with two icons, i.e. a paper clip and an ACROBAT icon and a paper clip and a PHOTOSHOP icon. The dual icon display indicates that the mail was created without any special authoring/reading component and has an attachment created by some other program, in this case ACROBAT and PHOTOSHOP.

As mentioned above, the electronic mail software of the present invention is described by example with reference to the email software of my previously incorporated parent application which includes a main email component and a number of installable components. According to the presently preferred embodiment, a component is also included for maintaining a database of icons. Figure 3 illustrates the relationship between these components.

As shown in Figure 3, a data structure 10 referred to as TYPETABLE is created and maintained by a component 12 referred to as TYPE_UPDATER. A working example of a TYPE_UPDATER according to the invention is illustrated in Appendix B which is described in more detail below. The primary purpose of the TYPETABLE is to be read by the MBOX_DISPLAYER component 14. The TYPE_UPDATER 12 and MBOX_DISPLAYER 14 communicate with each other using function calls and a shared data structure, TYPETABLE. For example, the MBOX_DISPLAYER includes a call to the function "initialize TYPETABLE" (306, Figure 6a) inside the TYPE_UPDATER component. The MBOX_DISPLAYER communicates with the electronic mail client software using the API described in the parent

application and or in previously incorporated serial number [BAK-007]. As shown in Figure 3 the MBOX_DISPLAYER component 14 and the electronic mail client software 16 also have bidirectional access to the store of electronic mail messages in the user's mail boxes. According to the presently preferred embodiment, the electronic mail messages are stored with an optional TYPE field which includes information drawn from the MIME type and subtype header fields of the message if the message is created by an installable application component of the kind described in the parent application.

The MBOX_DISPLAYER component functionality is invoked by the user when a mailbox is opened, when the list of mail is scrolled, etc. Those skilled in the art will appreciate that most electronic mail client software provides a number of different mailboxes such as inbox, outbox, read mail, sent mail, etc. The MBOX_DISPLAYER component 14 preferably includes all of the functionality of state-of-the-art mailbox displayers and also includes the functionality of looking to the TYPETABLE (and as explained in more detail below, to look into the body of a MIME message) to find an appropriate icon for display alongside a message title in the mailbox display. According to the presently preferred embodiment, icons are stored as small image files, e.g. EPS files or GIF files, and are pointed to by the TYPETABLE data structure. In order to accommodate the use of different size fonts in the mailbox display, means for scaling the size of the icon graphics are also provided. Three methods may be used. First, the image may be scaled using a standard interpolation algorithm. Second, multiple copies of icon images with different resolutions may be stored and retrieved to match a limited number of font point sizes. Third, and presently preferred, a combination of the first two methods is used whereby at least one image for each icon is stored, the icon most closely matching the point size of the font is chosen and then scaled as needed to better match the font point size.

Table 1 illustrates how the TYPETABLE data is stored in permanent storage (e.g. hard disk).

mimetype	icon filename	msg handler filename
text/plain	c:\kidcode\text.gif	c:\kidcode\txt.dxr
x-application/ rebus	c:\kidcode\rebus.gif	c:\kidcode\rebus.dxr
x-application/grid	c:\kidcode\grid.gif	c:\kidcode\grid.dxr
x-application/graph	c:\kidcode\graph.gif	c:\kidcode\grph.dxr
...	...	...
multipart/mixed	c:\kidcode\paperclip.gif	

Table 1

Table 1 illustrates at least five mimetypes. The first four are mimetypes which utilize installable components for authoring/reading. The installable components are indicated by the ".dxr" file extension. The multipart/mixed (fifth) mimetype illustrated in Table 1 indicates an attachment created with an external application rather than an installable component. As mentioned above with reference to Figure 2a, a generic paper clip icon is used to distinguish this attachment from mail messages created with installable components.

Table 2 illustrates the typetable data structure as it is loaded into RAM.

mimetype	ptr	icon filename	msg handler filename
text/plain	20	c:\kidcode\text.gif	c:\kidcode\txt.dxr
x-application/ rebus	21	c:\kidcode\rebus.gif	c:\kidcode\rebus.dxr
x-application/grid	22	c:\kidcode\grid.gif	c:\kidcode\grid.dxr
x-application/graph	23	c:\kidcode\graph.gif	c:\kidcode\grph.dxr
...		...	...
multipart/mixed	19	c:\kidcode\paperclip.gif	

Table 2

When the data structure TYPETABLE is loaded into RAM it is referred to as SG_TYPETABLE and has a structure as shown in Table 2. This structure includes a pointer to the icon. The pointer in the exemplary embodiment is a LINGO castmember.

As mentioned above, according to the exemplary embodiment, when the email message has an attached file, a generic attachment icon is displayed and an icon particular to the attachment is also displayed. This second icon is pointed to by a data structure in RAM which is created on the fly (e.g. as illustrated in Figure 6b) referred to as SG_ATTACH_TYPETABLE. The basic structure of SG_ATTACH_TYPETABLE is illustrated in Table 3.

File Extension	icon pointer	program filename
.doc	30	c:\programs\winword.exe
.pdf	31	c:\programs\acrobat.exe
.html	32	c:\programs\netscape.exe
.htm	32	c:\programs\netscape.exe
.xml	32	c:\programs\netscape.exe
...	...	...

Table 3

As shown in Table 3, the file extension is associated with an icon pointer and the pathname to the program which will be used to read the attachment. As discussed in more detail below with reference to Figure 6b, the SG_ATTACH_TYPETABLE is built on the fly using icons from the system registry.

A presently preferred embodiment of a MBOX_DISPLAYER component is presented in detail in Appendix A which is similar to Appendix B of my previously incorporated parent application. The code listing shown in Appendix A hereto differs from the code listing of the parent application starting at line 287 which is the start of the main mailbox display function. The main mailbox display function is also illustrated by the flowchart of Figure 4 Referring now to Figure 4 and Appendix A, the function starts at line 287 in Appendix A and at the START 101 in the flowchart of Figure 4. Before displaying the mailbox contents, pointers to the TYPETABLE and associated system features are set up as illustrated at lines 293-300 in Appendix A and at 103, 105 in Figure 4. The various mailbox fields, e.g. message number,

subject, date, message read indicator, are cleared at lines 302-307 in Appendix A and at 107 in Figure 4. "Sprite channels" (MACROMEDIA graphic holders) which will be used to display icon graphics are cleared at lines 309-311 in Appendix A and at 107 in Figure 4. Next the message list is displayed starting at line 313 in Appendix A and at 109 in Figure 4. The elements which make up each line of the mailbox display (e.g. the message subject, the date, the sender's name, as well as the graphical elements) are referred to as "properties". These properties are read from the message at lines 320-326 in Appendix A and at 111, 113 in Figure 4. With the exception of mailbox, mimetype, and status, all of the properties are automatically displayed when read as illustrated at 111 in Figure 4 and lines 320-323 in Appendix A. The mailbox, mimetype, and status properties are read at 113 in Figure 4 and lines 324-326 in Appendix A. According to the presently preferred embodiment, the mimetype icon is also used to display message status. If the message has been read, the icon is displayed in greyscale. If the message has not been read, the icon is displayed in color. The steps of finding the icon and setting it to greyscale or color are illustrated at lines 328-355 of Appendix A. The sprite channel counter is set at lines 332-334 in Appendix A and at 115 in Figure 4. The TYPETABLE data structure is parsed at lines 340-343 in Appendix A and at 117 in Figure 4. If the TYPETABLE does not contain the icon indicated by the mimetype for this message, then a default icon is chosen at lines 344-346 in Appendix A and at 119 in Figure 4. If the status for the message indicates that it (its attachment) has been read, then the greyscale version of the icon is set at lines 348-349 in Appendix A and at 121 in Figure 4. The chosen icon is added to an icon list for rapid access during scrolling of the mailbox contents list. This is illustrated at lines 351-352 in Appendix A and at 123 in Figure 4. The remainder of the main mailbox display function at lines 354-374 in Appendix A and at 125 in Figure 4 concern locating the text and icons at appropriate screen locations. The code shown at lines 315-374 repeats for the number of messages in the mailbox as illustrated by the decision at 127 in Figure 4. When there are no more messages to be listed, the main mailbox display function ends as illustrated at 376 in Appendix A and 129 in Figure 4.

Though not presently illustrated in the code or flowchart, as mentioned above, if there is no appropriate icon in the directory of icons, the mailbox displayer uses icon image data contained in a subpart of the MIME message if it is available. Those skilled in the art will appreciate that this functionality is easily performed by reading the icon image data from the location in the MIME file which is defined by the standard(s) referenced above. Code to implement this might be inserted at line 325 of Appendix A or at line 345 of Appendix A.

As mentioned above, the remainder of Appendix A is substantially the same as the mailbox component of Appendix B of the parent application and the description of it is adequately set forth in the previously incorporated parent application. Those skilled in the art will appreciate that the code listings of the Appendices are particular to the MACROMEDIA DIRECTOR development suite and that the same functionality may be achieved using a different development environment. Figure 5 illustrates the functionality of the mailbox display function in a more generic manner which can apply to different programming languages.

Referring now to Figure 5, the mailbox display starts at 201 to read the list of messages. It checks at 202 to determine whether the TYPETABLE has been initialized. If it has not, the TypeTable is initialized at 204 as described in detail below with reference to Figure 7. The mailbox display gets the next message listing at 203. For the message listing obtained at 203, the mailbox proceeds to obtain properties for the listing at 205 and property values at 207. If it is determined at 209 that the property uses an icon for display, a "get icon" routine is called at 211 (this routine is illustrated at Figure 6). The message properties that concern the present application are (a) the message MIME type and (b) whether the message has a file attachment. In the flow charts in Figure 5 (209) the system looks up whether current message property which is implemented as a LINGO symbol, e.g. #type, #date, #mailbox, #status, is represented by an icon. Although in the current implementation, each property is coded separately, the more general implementation described in the flowchart of Figure 5 could be accomplished by checking whether the target symbol, e.g. #type was an element of a list data structure, e.g. Properties_with_Icons = (#type, #has_attachment) in procedure 209 of Figure 5. The icon is displayed at 213. If it was determined at 209 that the property does not use an icon, the property value is displayed at 215. At 217 it is determined whether there are additional properties for this message listing. If there are, the program returns to 205. If there are not, it is determined at 219 whether there are more message listings to list. If there are, the program returns to 203. If there are not, the program ends at 221.

Figure 6 illustrates a generalization of the functionality contained in Appendix A at lines 328-350. This is the generalized "get icon" routine called at 211 in Figure 5. The routine starts at 301 having been provided the property name and property value by the calling program. If it is determined at 303 that the property is "mimetype" or, to conform to the LINGO implementation, "#type", the icon of the mimetype is obtained at 305 and a pointer to the icon is passed back to the calling program at 307. The actual steps involved in getting the icon for a mimetype are illustrated in Figure 6a. If it is determined at 309 that the property is "has attachment", the icon for "has attachment" is obtained at 311 and a pointer to the icon is passed

back to the calling program at 307. The procedure that gets the file attachment icon is described in Figure 6b for a Microsoft Windows platform. Alternatively, the TYPETABLE data structure could include a field for filetype that is used to map from file extensions to MIME types and MIME type icons. If TYPEDATA were modified in this way, both message type and file type icon lookups would use the TYPETABLE. A related modification in the TYPE_UPDATER would be required to install the filetype/MIMEtype associations as they are encountered either via a previously unknown message type or a previously unknown file type. Because file type extensions are not as rich as Mimetypes, the same file type extension may map to many different MIME types. For example this could occur for different versions of the same software if the software manufacturer assigns different MIME subtypes for different versions of their software.

If it is determined at 313 that the property is "message read", the icon for "message read" is obtained at 315 and a pointer to the icon is passed back to the calling program at 307. If it is determined at 317 that the property is "priority", the icon for "priority" is obtained at 319 and a pointer to the icon is passed back to the calling program at 307. From the foregoing, those skilled in the art will appreciate that many different icons can be displayed for different properties.

Figure 6a illustrates a generalization of the implementation contained at lines 328-346 of Appendix A. This is the "get icon for mimetype" function called at 305 in Figure 6. This routine starts at 302 having been given the "mimetype" by the calling program. It determines at 304 whether the TYPETABLE has been initialized. If not, initialization is performed at 306. The initialization routine may reside in the TYPE_UPDATER component or may call functions that reside in the TYPE_UPDATER component. For example, in the LINGO implementation, the function Read_TypeTable_File (Figure 8), which is part of the TYPE_UPDATER component is used to initialize the TYPETABLE data structure. After initialization, or if it was determined at 306 that the TYPETABLE was already initialized, the mimetype is used to retrieve an icon pointer from the TYPETABLE at 308. See Appendix A, lines 328-342. It is determined at 310 whether the icon pointer is null. If it is, an "install message type" routine is called at 312. The "install message type" routine, which is contained in the TYPE_UPDATER component, is explained in detail below with reference to Figure 11which illustrates the installation of new message handlers and icons for mimetypes. If the icon pointer is not null, it is determined at 314 whether the icon pointer points to "icon not installed". If that is the case, the icon pointer is set to the default pointer at 316. In either case, a non-null pointer is returned to the calling program at 318.

Figure 6b illustrates an exemplary procedure for getting an icon for an attachment to an email message. Starting at 320, the file extension of the attachment is read. (Note that on a Windows platform, the three letters following "." in a filename determine the "filetype". With other platforms, such as the Macintosh platform, the filetype and "creator code" are listed in the "resource fork" of the file. Thus, for those platforms, the first step will be to read the filetype (and creator code) from the resource fork of the file.) Once the filetype (or filetype and creator code) have been determined, the routine attempts at 322 to find an appropriate icon in the SG_TYPETABLE. If it is determined at 324 that no appropriate icon has been found, the routine attempts at 326 to find an appropriate icon in the SG_ATTACH_TYPETABLE. If it is determined at 328 that no appropriate icon has been found, the routine attempts at 330 and 332 to find an appropriate icon in the system registry. (Note that with other operating systems, icon resources may be stored in different places. E.g., in the Macintosh OS, icon resources are stored in the invisible "desktop" file.) If at 334 a suitable icon is found, a pointer to the icon is set at 336 and the pointer is written to the SG_ATTACH_TYPETABLE at 338. The icon pointer is returned to the mailbox displayer at 340. See 311 in Figure 6 and 211 in Figure 5. If a suitable icon is not found at 342, no icon pointer is provided.

Turning now to Appendix B, lines 1-26 provide an overview and introduction to the Type Updater. The Type updater includes eleven functions. Three of the them are public functions called by the mailbox displayer. These include: Initialize_TypeTable, Install_Type, and Uninstall_Type. The remaining eight functions are private functions used within the Type Updater. These include: Write_Typetable_File, Read_Typetable_File, Read_Icon_Files_To_RAM, read_iconFile, delete_mimetype, insert_mimetype, delete_filetype, insert_filetype. The Initialize_TypeTable function is illustrated in Figure 7 and at lines 29-60 in Appendix B. The function begins at 360 in Figure 7, sets the SG_TYPETABLE to nil at 362 (line 39 in Appendix B). The SG_ATTACH_TYPETABLE is set to nil at 364 (line 43 in Appendix B). The Typetable file stored on disk is read into SG_TYPETABLE at 366 (line 44 in Appendix B). The read_typetable_file internal function is illustrated in Figure 8 and at lines 207-252 in Appendix B. An error check is performed at 368 (line 46 in Appendix B) to determine whether any data was loaded into RAM. If no data was read, the user is alerted at 370 (line 47 in Appendix B) and a failure is returned at 372 (line 48 in Appendix B). Otherwise, icon files are read into RAM at 374 (line 51 in Appendix B). An error check is performed at 376 (line 47 in Appendix B). If the data was read successfully, the function ends at 378 (line 60 in Appendix B).

Turning now to Figure 8 and at lines 207-252 in Appendix B, the read_typetable_file function starts at 380 (line 218 in Appendix B). The SG_TYPETABLE is set to nil at 382 (line 222 in Appendix B). The typetable.txt file is opened at 384 (lines 224-227 in Appendix B) and a file open error check is performed at 386 (line 229 in Appendix B). If an error in opening the file is detected, an error alert is produced at 388 and (line 230 in Appendix B), the file is closed at 390 (line 231 in Appendix B), and a SG_TYPETABLE is returned at 392 (line 232 in Appendix B). If the file was opened without error, the first line is read at 391 which begins a loop (steps 394-408) which ultimately ends at 394 when the end of file is reached after which the file is closed at 390 (line 231 in Appendix B), and a SG_TYPETABLE is returned at 392 (line 232 in Appendix B). As shown in Appendix B at lines 235-239, the LINGO implementation reads the entire file into a string and simulates line by line reading because LINGO cannot read line by line. When a line is read, the first word in the line is the mimetype (line 243 in Appendix B). The mimetype is inserted into SG_TYPETABLE at 396 (line 244 in Appendix B). If the mimetype is not defined as discovered at 398, the user is alerted at 400 and the next line is read at 391. If the mimetype is defined, the next word in the line (filetype) is read and inserted into SG_TYPETABLE at 402 (line 245 in Appendix B). The next word in the line (iconFileName) is read and inserted into SG_TYPETABLE at 404 (line 246 in Appendix B). The next word in the line (msgHandler) is read and inserted into SG_TYPETABLE at 406 (line 247 in Appendix B). Any errors detected at 408 are reported at 400. As mentioned above, the process continues until the typetable.txt file is completely read.

The function Write_TypeTable_File is described in Figure 9 and at lines 159-206 of Appendix B. This function writes the contents of SG_TYPETABLE back to typetable.txt after new mimetypes and icons have been added to the SG_TYPETABLE via the Install_Type function described below with reference to Figure 11. The Write_TypeTable_File function begins at 410 (line 163 in Appendix B) and first checks at 412 (line 168 in Appendix B) whether the SG_TYPETABLE structure is empty. If it is empty, it returns an error message at 414 (line 169 in Appendix B) and a failure message at 416 (line 170 in Appendix B). If the SG_TYPETABLE structure is not empty a backup copy of the typetable.txt file is created at 418 (line 173 in Appendix B) and a new empty file is created at 420 (lines 176-180 in Appendix B). An error check is performed at 422 (line 181 in Appendix B) and if an error is detected in creating the new file, an error message is returned at 424 (line 182 in Appendix B). The backup file is restored at 426 (line 183 in Appendix B) and a return code is set to "fail" at 428 (line 184 in Appendix B). The typetable.txt file is closed at 430 (line 203 in Appendix B). The backup is deleted at 432 (line 204 in Appendix B) and the return code is returned at 434 (lines 205-206 in Appendix B). If there is no error creating the new file, it is opened for write access at line 187-

188 in Appendix B and a starting line counter is set at line 191. The first (next) entry in the SG_TYPETABLE is read at 436 (line 192 in Appendix B). If it is determined at 438 (line 193 in Appendix B) that the end of SG_TYPETABLE has been reached, the return code is set to "success" at 440 (line 201 in Appendix B), the file is closed 430, the backup is deleted 432 and the return code is returned 434.

Until the end of SG_TYPETABLE is reached, data is set to write at 442 and the mimetype is written at 446 (line 194 in Appendix B) to a string. The filetype, iconfilename, and messagehandlerfilename are added to the string at 446, 448, and 450 respectively (lines 195-197 in Appendix B). The string is written to the new typetable.txt file at 452 (line 198 in Appendix B) and a line delimiter is written at 454 (lines 199-200 in Appendix B). The function loops back to 436 and continues until all of the entries in SG_TYPETABLE are read and written to the new typetable.txt file.

Figure 10 illustrates the function Read_Icon_Files_To_RAM which is presented in Appendix B at lines 255-290. Starting at 456 in Figure 10 (line 255 in Appendix B), the function first determines at 458 (line 259 in Appendix B) whether any mimetypes are defined in the structure SG_TYPETABLE. If there are none, no icons are defined, an error message is returned at 460 (line 260 in Appendix B), the return code is set to "fail" at 462 (line 261 in Appendix B) and the return code is returned at 464 (lines 261-262 in Appendix B). If the structure SG_TYPETABLE is not empty, the function loads the default icon into RAM and sets a pointer to it at 466 (line 265 in Appendix B). Icon pointer numbers are related to LINGO castNumbers at line 268 in Appendix B and a counter for incrementing the castNumbers is set at line 270. The first (next) entry in the structure SG_TYPETABLE is read at 468 (line 271 in Appendix B). If it is determined at 470 (line 272 in Appendix B) that there are no more entries to read, the return code is set to "success" at 472 and the return code is returned at 464 (line 288 in Appendix B). So long as entries remain, the mimetype is read at 474 and the iconfilename is for the mimetype is read at 476 (line 273 in Appendix B). If it is determined at 478 (line 274 in Appendix B) that there is no icon associated with this mimetype, the default icon pointer is assigned to it at 480 (line 275 in Appendix B). Otherwise, the next icon bitmap and pointer are read at 482 (line 277 in Appendix B) using the function read_iconfile (lines 295-298 in Appendix B). If the icon pointer is not nil as determined at 484 (line 278 in Appendix B), the castNum is incremented at line 279. Otherwise the castNum is not incremented and the default icon pointer is used at 480 (line 280 in Appendix B). In either case, the icon pointer associated with the mimetype is inserted at 486 (line 283 in Appendix B). The counter is incremented at line 284 and the next entry from SG_TYPETABLE is read at 468 (line 285 in Appendix B).

Figure 11 illustrates the function install_type which is used to install a new icon and/or message handler for a particular mimetype or filetype. The function starts at 600 with input which includes an icon file name, a message handler program name, and either a mimetype or a filetype (lines 71-75 in Appendix B). This input can be provided by the user or by an automatic means as described below. The typetable file is read into RAM at 602 (line 78 in Appendix B) thereby creating SG_TYPETABLE. If SG_TYPETABLE is empty as determined at 604 (line 80 in Appendix B), an error is returned at 606 (lines 81-82 in Appendix B). Otherwise, the mimetype (or filetype) specified at 600 is retrieved at 608 (line 87 in Appendix B) from SG_TYPETABLE if it exists. If the mimetype (or filetype) exists in SG_TYPETABLE and if it is already associated with an icon and message handler, the user is prompted at 612 (line 90 in Appendix B) whether it should be redefined. If the user chooses NO, an error is returned at 614 (line 94 in Appendix B). If the mimetype was not previously defined or if the user chooses to redefine it, the message handler program specified at 600 is checked for validity (e.g. whether it is present on the hard disk or network) at 616 (line 102 in Appendix B). If it is not valid, an error is returned at 614 (lines 104-105 in Appendix B). If the message handler and the mimetype (filetype) are valid, they are associated with each other in SG_TYPETABLE at 618 (line 108 in Appendix B). Next, the icon file name specified at 600 is checked for validity at 620 (line 110 in Appendix B). If it is not valid, the default icon is specified at 622 (line 112 in Appendix B) and an error alert is returned at 624 (line 111 in Appendix B). In either case, the icon file name specified or the default icon, if necessary, is associated with the message handler and the mimetype (filetype) in SG_TYPETABLE at 626 (line 114 in Appendix B). The typetable.txt file is written back from SG_TYPETABLE at 628 (lines 122-125 in Appendix B) and unless an error is encountered, a success is returned at 630 (line 126 in Appendix B).

Those skilled in the art will appreciate that the install_type function can be called by another program so that icons are automatically installed/removed when modular authoring/reading components are installed/removed. Though not specifically shown in code or drawings herein, those skilled in the art will appreciate how to implement this second embodiment by referring to Appendix B herein and the component installing code of the parent application.

According to a third embodiment, new icons are added automatically whenever a new message type is encountered by the mailbox displayer. The new icon is retrieved from either the operating system registry or from the icon image data embedded in the message. Those skilled in the art will appreciate the implementation of this embodiment by reference to Appendix B herein and the above referenced MIME standards.

According to a fourth embodiment, the type updater automatically queries a network server for new icon information and downloads icon image data as needed or as scheduled. Those skilled in the art will appreciate that automatic updaters which download data from a file server are well known and that this embodiment may be implemented by reference to the known automatic updaters together with Appendix B herein.

There have been described and illustrated herein several embodiments of an enhanced electronic mail system including methods and apparatus for identifying mime types and for displaying different icons. While particular embodiments of the invention have been described, it is not intended that the invention be limited thereto, as it is intended that the invention be as broad in scope as the art will allow and that the specification be read likewise. Thus, while particular code listings have been disclosed, it will be appreciated that other code could be utilized. For example, although the Appendices illustrate the invention with MACROMEDIA DIRECTOR LINGO code, the invention could be embodied with C++, based on a "MOZILLA" engine, or via a number of other popular development tools. Also, while the invention has been disclosed with reference to the applicant's parent application which involves an email client having installable authoring/reading components, it will be recognized that the principles disclosed herein for displaying icons representing mimetypes in a mailbox listing may be applied to other kinds of email clients. Moreover, while particular configurations have been disclosed in reference to the way icon images are stored (i.e. scalable images), it will be appreciated that other configurations could be used as well. Further, while the invention has been shown with reference to display on a computer screen, it will be appreciated that the display may be on a television, a personal digital assistant, a cell phone, a wrist watch, etc. It will therefore be appreciated by those skilled in the art that yet other modifications could be made to the provided invention without deviating from its spirit and scope as so claimed.

Appendix A - Mailbox Displayer Code - Page 1

```
— MBOX_DISPLAYER IMPLEMENTATION
—— Code for a mailbox handler for the KidCode electronic mail client.
—— This Director MIAW displays a mailbox in a window as a list
— of messages with one line for each message. Each message has the following fields displayed:
— 1. message number
— 2. Message sender
— 3. Message mimetype & status (single icon used to indicate both properties)
— 4. Message subject header
— 5. Message date


on startMovie
	global SG_lastActiveWindow — super global that keeps track of KC last active window
	global mbxG_username — current user name
	global mbxG_messages — list of messages
	global mbxG_nMsgs — number of messages in mailbox
	global mbxG_boxName — current mailbox name
	global mbxG_whichLine — current hilite line = msgNumber
	global mbxG_subtractLine
	global mbxG_lips

	set mbxG_lips = 0

	— called by API to Main Movie
	tell the stage to emh_continue(#mailbox)
end


on activateWindow
	global SG_lastActiveWindow
	global mbxG_myWindow
	set SG_lastActiveWindow = mbxG_myWindow
end activateWindow


— Stop movie handlers
-----------------------------------------------------------------
StopMovie handler in a MIAW is called only when the movie
— plays through to the end or jumps to another movie.
— It isn't called when the window is closed or the window
— is deleted by the forget window command.

on stopMovie
	cleanUpMovie()
end
-----------------------------------------------------------------
called to close a MIAW or automatically whenever
— forgetWindow is called

on closeWindow
	cleanUpMovie()
```

Appendix A - Mailbox Displayer Code - Page 2

```
52  end
53  ------------------------------------------------------------
54
55  -- cleanUpMovie can be called by both stopMovie and
56  -- closeWindow.
57
58  on cleanUpMovie
59          global mbxG_whichLine
60          global mbxG_subtractLine
61          global mbxG_nMsgs
62          global mbxG_username
63          global mbxG_lips
64
65          -- CLEAR FIELDS AND GLOBAL VARIABLES
66
67          put "" into field "MailboxTitle"
68          put "" into field "prepositionTitle"
69          put "" into field "MessageNumber"
70          put "" into field "MailboxTo"
71          put "" into field "MailboxSubject"
72          put "" into field "MailboxDate"
73          put "" into field "Messageread"
74
75          set mbxG_nMsgs = 0
76          set mbxG_lips = 0
77          resetHilite()
78
79          if findEmpty(member 50) > 50 then
80                  set the scriptText of member 50 = ""
81          end if
82
83          set the memberNum of sprite 6 = the memberNum of member "lips up"
84          set the loc of sprite 4 to point(800, 4)
85          set the loc of sprite 5 = point(800, 19)
86  end cleanUpMovie
87
88
89  -- API Public Handlers ------------------------------------------------------
90  --
91  -- Ugly hack to work around problem with Director startup
92  -- of MIAWs. The problem is that, after calling a handler in the
93  -- MIAW, the StartMovie handler for the MIAW does not run until
94  -- the calling movie advances to its next frame.
95  -- Therefore, the calling sequence in the calling movie
96  -- has to be engineered so that the real handlers in the MIAW do not
97  -- run until after control has been transfered back to the calling
98  -- movie. However, at least one handler in the MIAW must be called
99  -- by the calling movie before the StartMovie handler will run.
100
101 -- startMeUp is the fake handler that, when called by the
102 -- main movie, will upon return to the main movie,
```

Appendix A - Mailbox Displayer Code - Page 3

```
103  — cause this movie's startMovie handler to run.
104
105  — The second part of this wormy hack is contained in the MIAW's
106  — startMovie handler... It is a call to a workAround handler in
107  — the calling movie called continueComponent
108  — The calling movie's continueRebus handler calls the real handlers
109  — in the MIAW.
110
111  on emc_startMeUp
112          — put "Macromedia sucks!"
113          return(TRUE)
114  end emc_startMeUp
115
116  ------------------------------------------------------------------------
117  initWindow is called by email main when a message handler
118  — is opened
119
120  on emc_initWindow userName, windowName
121          global mbxG_myWindow
122          global mbxG_username
123          global mbxG_platformType
124
125  — put "ENTER emc_initWindow mailbox"
126          set mbxG_username = userName
127          set mbxG_myWindow = windowName
128
129          — puppet the hilite (MB 4-17-99) removed this
130          resetHilite()
131          tell the stage to emh_getColorDepth()
132          set colorDepth = the result
133          mapColors(colorDepth)
134          tell the stage to emh_getPlatformType()
135          set mbxG_platformType = the result
136
137  — put "EXIT emc_initWindow mailbox"
138
139          return(TRUE)
140
141  end emc_initWindow
142
143  ------------------------------------------------------------------------
144  — closeWindow is not called unless Rebus plays as
145  — a MIAW.
146
147  on emc_closeWindow
148  — put "ENTER emc_closeWindow Mailbox"
149          closeWindow()
150          — step frame
151  — put "EXIT emc_closeWindow Mailbox"
152          return(TRUE)
153
```

Appendix A - Mailbox Displayer Code - Page 4

```
154  end emc_closeWindow
155
156  ------------------------------------------------------------------------
157
158  on emc_getComponentInfo
159          return( list( "SimpleMail", 1, #mailbox, "text" ) )
160  end emc_getComponentInfo
161
162  ------------------------------------------------------------------------
163
164  on mbx_getMessage
165  --
166  -- -- "open" button and doubleClick of highlighted message
167  -- -- calls to email main to hand the message
168  -- -- selected to a message handling movie
169  -- -- This script was previously the "open" cast memeber script:
170  --
171  -- global mbxG_whichLine
172  -- global mbxG_messages
173  --
174  -- set mailData = getAt(mbxG_messages, mbxG_whichLine)
175  --
176  -- return(mailData)
177  --
178  --
179  --
180  --end mbx_getMessage
181
182  ------------------------------------------------------------------------
183
184  on mbx_getMessageNumber
185          global mbxG_whichLine
186          return(mbxG_whichLine)
187  end mbx_getMessageNumber
188
189  ------------------------------------------------------------------------
190  -- mbx_trashMessages returns a list of messages that are to
191  -- be trashed in the mailfile. Email main will rewrite the mail file
192  -- When implemented correctly, it will determine which message numbers
193  -- are associated with the currently selected lines in the mailbox
194  -- display, update the display to remove these messages from the
195  -- list, and return the list of deleted message numbers.
196
197  on mbx_trashMessages
198
199          global mbxG_messages -- list of messages
200          global mbxG_nMsgs -- number of messages in mailbox
201          global mbxG_whichLine -- current hilite line = msgNumber
202
203          -- set mailData = getAt(mbxG_messages, mbxG_whichLine)
204          --
```

Appendix A - Mailbox Displayer Code - Page 5

```
205         — — tell the stage
206         — return(mailData)
207         — — end tell
208
209         — needs implementation that can handle multiple messages
210         — also need to rewrite trashIt which does not conform to
211         — API rule that only API handlers can be called in other movies
212         —
213         — set message = mbxG_messages
214         if mbxG_whichLine > 0 AND mbxG_whichLine <= mbxG_nMsgs then
215                 tell the stage to emh_alertUserToTrash()
216                 set yes = the result
217                 if not yes then return [] -- user canceled
218
219                 set trashList = list(getAt(mbxG_messages, mbxG_whichLine))
220                 deleteAt(mbxG_messages, mbxG_whichLine)
221                 set mbxG_nMsgs = mbxG_nMsgs - 1
222                 displayMailbox(mbxG_messages)
223                 resetHilite()
224         else
225                 alert("Please click on the message you wish to delete.")
226                 set trashList = []
227         end if
228
229         return(trashlist)
230
231   end mbx_trashMessages
232
233   ---------------------------------------------------------------------
234   — accepts a mailbox datastructure that consists of a boxname and
235   — a list of messages
236
237   on mbx_openMailbox mailbox
238         global mbxG_username
239         global mbxG_messages
240         global mbxG_boxName
241         global mbxG_nMsgs
242
243   — put "ENTER mbx_openMailbox"
244         set mbxG_boxName = getAt(mailbox, 1)
245         put mbxG_userName & "'s " & mbxG_boxName into field "mailboxTitle"
246
247         set mbxG_messages = getAt(mailbox, 2)
248         set mbxG_nMsgs = count(mbxG_messages)
249
250         displayMailbox(mbxG_messages)
251
252   — put "EXIT mbx_openMailbox"
253         return(TRUE)
254
255   end mbx_openMailbox
```

Appendix A - Mailbox Displayer Code - Page 6

```
256
257   ---------------------------------------------------------------
258
259   Utilities –
260   ---------------------------------------------------------------
261   Initialize formatting of text fields
262   — Thanks to Frank Leahy for this one
263
264   on SetTextInfo fldName, fldValue, fldAlign, fldFont, fldSize, fldStyle
265         if fldValue <> EMPTY then
266               put fldValue into field fldName
267         end if
268         set the textAlign of field fldName = fldAlign
269         set the textFont of field fldName = fldFont
270         set the textSize of field fldName = fldSize
271         set the textStyle of field fldName = fldStyle
272   end
273
274   ---------------------------------------------------------------
275   on formatFields
276
277         — FORMAT THE TEXT FIELDS
278
279         setTextInfo "MessageNumber", " ", "left", "arial", 14, "bold"
280         setTextInfo "MailboxTo", " ", "left", "arial", 14, "bold"
281         setTextInfo "MailboxSubject", " ", "left", "arial", 14, "bold"
282         setTextInfo "MailboxDate", " ", "left", "arial", 14, "bold"
283         setTextInfo "Messageread", " ", "left", "arial", 14, "bold"
284
285   end formatFields
286
287   — MAIN MAILBOX DISPLAY FUNCTION
288   — displays a Mailbox style listing of messages
289   -- places the appropriate components from each message
290   — into field members with lines aligned for display
291
292   on displayMailBox msgList
293         global SG_TYPETABLE — super global variable shared across different MIAWS
294         global mbxG_red
295         global mbxG_platformType
296         global mbxG_iconList
297
298         — mbxG_iconList for future use in up/down scroll scripts
299         set mbxG_iconList = [:]
300         set count = 0
301
302         — first clear all the fields and the sprites
303         put "" into field "MessageNumber"
304         put "" into field "MailboxTo"
305         put "" into field "mailboxSubject"
306         put "" into field "mailboxDate"
```

Appendix A - Mailbox Displayer Code - Page 7

```
        put "" into field "Messageread"

        repeat with i = 40 to 70
                set the member of sprite i = member "blank"
        end repeat

        -- Fill the text fields with info from all of the messages so that it is available
        -- when the window scrolls
        repeat with msg in msgList

        ---- with the exception of mailbox, mimetype and status,
        ---- the fields will automatically be displayed when filled

        put the lineCount of member "MailboxTo"+1 & RETURN after field "MessageNumber"
        put getProp(msg, #from) & RETURN after field "MailboxTo"
        put getProp(msg, #subject) & RETURN after field "mailboxSubject"
        put getProp(msg, #date) & RETURN after field "mailboxDate"
        put getProp(msg, #mailbox) into mailbox
        put getProp(msg, #mimetype) into mime
        put getProp(msg, #status) into status

        ---- Display the mimetype icon
        ---- We use the mimetype icon to also indicate message status. A greyscale version of the
        -- icon is displayed if a message has been read. Otherwise a color icon is displayed.

        ---- Icons will be placed in sprite channel 40 and beyond
                set i = 40 + count
                set count = count + 1

        -- find the icon for the message mimetype
        ---- this code should be rewritten using a data access function. For now we need to
        ---- know the format of the TYPETABLE data structure

                set mimeProperties = getProp(SG_TYPETABLE, mime)
                set iconCastMember = getAt(mimeProperties, 2) -- second item is the cast member
                number

        -- if this type is unknown then use the default icon
                if iconCastMember = 0 then set iconCastMember = the number of member
                "DefaultIcon"

        -- get the greyscale version of the icon if the message has been read.
                if status = "R" then set iconCastMember = iconCastMember + 1

        -- add the icon to the list of icons used for scrolling the window
                append (mbx_iconList, iconCastMember)

        -- place the icon at the proper location for display
                set the memberNum of sprite i to iconCastMember

        -- display the icon in the correct grid cell in the mailbox message list.
```

Appendix A - Mailbox Displayer Code - Page 8

```
358             — set the locH of all icons to 50
359             set whereGoesIcon = the lineCount of member "MailboxTo"
360             puppetSprite i, TRUE
361             set the visible of sprite i = TRUE
362             set the locH of sprite i to 50
363
364             — precise placement of the icon next to it's message...
365             set positonVar = 105 + linePosToLocV(member "MailboxTo", whereGoesIcon)
366             set the locV of sprite i to positonVar
367             — but don't let icons fall go beyond window if there are many messages
368             if positonVar > 550 or positonVar < 105 then
369                     set the visible of sprite i = FALSE
370             else
371                     set the visible of sprite i = TRUE
372             end if
373             addProp mbxG_iconList, (the locV of sprite i), mime
374             end repeat
375
376     end displayMailbox
377
378     — FUNCTIONS USED TO RESPOND TO USER INTERACTION WITH MAILBOX
379     — HILITE MESSAGE is called when a user clicks a mouse on a message line
380
381     on hiliteMessage whichLine
382             global mbxG_nMsgs, mbxG_whichLine, mbxG_subtractLine, mbxG_messages
383
384     — KEEP TRACK OF SELECTED LINE
385
386     set mbxG_whichLine = whichLine
387
388     — MAKE SURE LINE IS VALID
389
390     if mbxG_whichLine <= 0 then
391             return(0) — do nothing, errors are caught elsewhere
392     else if mbxG_whichLine > mbxG_nMsgs then
393             — user clicked somewhere else in field
394             set mbxG_whichLine = 0 — reset to 0
395             return(0)
396     end if
397
398     — HIGHLIGHT SELECTED LINE
399
400     set whichHighlight = mbxG_whichLine + mbxG_subtractLine
401
402     — since all field members in display are kept synchronized
403     — any one will do for linePosToLocV
404     — use "MailboxTo", it's small
405
406     set the locV of Sprite 11 to ¬
407             (99 + linePosToLocV(member "MailboxTo", whichHighlight))
408
```

Appendix A - Mailbox Displayer Code - Page 9

```
--when the user double clicks on a hilited message, go get it from mailFile

if the doubleClick then
	set maildata = getAt(mbxG_messages, mbxG_whichLine)
	tell the stage
		emh_openMessage(maildata)
	end tell
	-- mbx_getMessage()
end if

end hiliteMessage


-----------------------------------------------------------------------

on resetHilite
	global mbxG_whichLine, mbx_subtractLine

	set mbxG_whichLine = 0
	set mbxG_subtractLine = 0

	-- SET HIGHLIGHT OFF STAGE
	set the loc of sprite 11 to point (11, -20)

end resetHilite
-----------------------------------------------------------------------
-- this is a lookup table for color
-- only really necessary for Mac platform
-- use on any color that you want to
-- set the forecolor of field

on mapColors colorDepth
	global mbxG_red
	global mbxG_blue
	global mbxG_white
	global mbxG_black

	case colorDepth of

		8:
			set mbxG_red = 6
			set mbxG_blue = 4
			set mbxG_white = 0
		16:
			set mbxG_red = 31744
			set mbxG_blue = 31
			set mbxG_white = 32767
		32:
			set mbxG_red = 16711680
			set mbxG_blue = 255
			set mbxG_white = 16777215
```

Appendix A - Mailbox Displayer Code - Page 10

```
460         end case
461
462         set mbxG_black = the forecolor of line 1 of member the member of sprite 4
463
464   end mapColors
465
466   ---------------------------------------------------------------------------
467   scripts run when the mouse is clicked on a mailbox message line.
468   A script is needed for each field in the message line.
469
470   on mouseUp
471         hiliteMessage (the clickon - 40)
472   end
473
474
475   on mouseDown
476
477         global mbxG_lips
478         set whichLine = the mouseLine
479         if mbxG_lips then
480               set astr = line whichline of field "MailboxTo"
481               speak(astr)
482         else
483               hiliteMessage(whichLine)
484         end if
485   end
486
487   on mouseDown
488
489         global mbxG_lips
490         set whichLine = the mouseLine
491         if mbxG_lips then
492               set astr = line whichline of field "MailboxSubject"
493               speak(astr)
494         else
495               hiliteMessage(whichLine)
496         end if
497   end
498
499   ---------------------------------------------------------------------------
500   more scripts run when the mouse is clicked on a mailbox message line
501
502   on mouseDown
503         global mbxG_lips
504         set whichLine = the mouseLine
505         if mbxG_lips then
506
507               set astr = line whichline of field "MailboxDate"
508               set aday = word 1 of astr
509               case aday of
510                     "Mon,": put "Monday" into word 1 of astr
```

Appendix A - Mailbox Displayer Code - Page 11

```
                        "Tue,": put "Tuesday" into word 1 of astr
                        "Wed,": put "Wednesday" into word 1 of astr
                        "Thu,": put "Thursday" into word 1 of astr
                        "Fri,": put "Friday" into word 1 of astr
                        "Sat,": put "Saturday" into word 1 of astr
                        "Sun,": put "Sunday" into word 1 of astr
                        otherwise
                end case

                speak(astr)
        else
                hiliteMessage(whichLine)
        end if
end

on mouseDown
        global mbxG_lips

        set whichLine = the mouseLine
        if mbxG_lips then
                set astr = line whichline of field "MessageNumber"
                speak(astr)
        else
                hiliteMessage(whichLine)
        end if
end


-- this script is attached to the message type icon which is displayed in the message line

on mouseUp
        set whichLine = the mouseLine
        hiliteMessage(mouseLine)
end

---- code for scroll buttons

on mouseDown
        global mbxG_whichLine
        global mbxG_subtractLine
        global mbxG_iconList

        -- SCROLL UP WITH HIGHLIGHT
        -- ICONS NOW SCROLL... HOWEVER, ICON SPRITE POSITION IS BASED ON
        -- MBXG_SUBTRACTLINE, NOT ON THE THE ACTUAL CORRESPONDING LINE
        NUMBER
        -- OF THE MESSAGE IN THE MAILBOX WINDOW.

        set numberOfIconsVar = count(mbxG_iconList)
        set lastIconPos = getPropAt(mbxG_iconList, count(mbxG_iconList))
        if lastIconPos >= 550 then
```

Appendix A - Mailbox Displayer Code - Page 12

```
562                 — there are enough messages to make scrolling nessisary
563                 repeat while the mouseDown = TRUE
564
565                 —oldSubtractLine gets set to mbxG_subtractLine before mbxG_sub. gets
566                 —incremented. This keeps the icons from falling one position behind
567                 —it's prospective message
568
569                 set oldSubtractLine = mbxG_subtractLine
570                 set mbxG_subtractLine = mbxG_subtractLine + 1
571
572                 if mbxG_subtractLine > 0 then
573                         set mbxG_subtractLine = 0
574                 end if
575
576         — SCROLL ALL FIELDS TOGETHER
577
578         scrollByLine member "MessageNumber", -1
579         scrollByLine member "MailboxTo", -1
580         scrollByLine member "MailboxSubject", -1
581         scrollByLine member "MailboxDate", -1
582         scrollByLine member "mime", -1
583         scrollByLine member "Messageread", -1
584
585         set numberOfIcons = the lineCount of member "MailboxTo" + 40
586         set amountOfMail = the lineCount of member "MailboxTo"
587
588         — for debugging
589         — put "linecount:" & the lineCount of member "MailboxTo"
590         — put "subtractline:" & mbxG_subtractLine
591
592         repeat with i = 40 to numberOfIcons
593
594
595                 — if the following 2 conditions are true, then scroll the icons
596                 — i.e. if messages scroll, icons do too, if not, then neither do icons.
597
598                 if mbxG_subtractLine >= - amountOfMail + 1 and oldSubtractLine < 0 then
599                         set the locV of sprite i = the locV of sprite i + 15
600                         if the locV of sprite i < 105 then
601                                 set the visible of sprite i = FALSE
602                         else
603                                 set the visible of sprite i = TRUE
604                         end if
605                 else
606                         nothing
607                 end if
608         end repeat
609
610         — MOVE HIGHLIGHT WITH LINE, MOVING HIGHLIGHT
611         — OFF SCREEN WHEN LINE MOVES OFF SCREEN
612         set whichHighlight = mbxG_whichLine + mbxG_subtractLine
```

Appendix A - Mailbox Displayer Code - Page 13

```
613         if whichHighlight <= 0 or whichHighlight >= 22 then
614         set the loc of sprite 11 to point (11, -20)
615         else — set the locV of highlight to scrolled message
616         set the locV of Sprite 11 to (99 + linePosToLocV(member "MailboxTo", whichHighlight))
617         end if
618         updateStage
619         end repeat
620         end if
621   end
622
623   on mouseUp
624         set numberOfIcons = the lineCount of member "MailboxTo" + 40
625         repeat with i = 40 to numberOfIcons
626               if the locV of sprite i > 550 or the locV of sprite i < 105 then
627                     set the visible of sprite i = FALSE
628               else
629                     set the visible of sprite i = TRUE
630               end if
631         end repeat
632   end
633
634   on mouseDown
635         global mbxG_whichLine
636         global mbxG_subtractLine
637         global mbxG_iconList
638
639
640         — AS MESSAGES ARE SCROLLED, ICONS NEED TO MOVE WITH THE
641         MESSAGE, AND
642         THE MEMBERNUM
643         — OF ICON SPRITES BE ASSIGNED TO THE NEW MESSAGE THAT IS VISIBLE
644         AFTER
645         — BEING SCROLLED.
646         — ICONS NOW SCROLL... HOWEVER, ICON SPRITE POSITION IS BASED ON
647         — MBXG_SUBTRACTLINE, NOT ON THE THE ACTUAL CORRESPONDING LINE
648         NUMBER
649         — OF THE MESSAGE IN THE MAILBOX WINDOW.
650         set numberOfIconsVar = count(mbxG_iconList)
651         set lastIconPos = getPropAt(mbxG_iconList, count(mbxG_iconList))
652         if lastIconPos >= 550 then
653               — there are enough messages to make scrolling nessisary
654               repeat while the mouseDown = TRUE
655                     scrollByLine member "MessageNumber", 1
656                     scrollByLine member "MailboxTo", 1
657                     scrollByLine member "MailboxSubject", 1
658                     scrollByLine member "MailboxDate", 1
659                     scrollByLine member "mime", 1
660                     scrollByLine member "Messageread", 1
661                     — GET NUMBER USED TO CORRECT FOR DISCREPANCY
662                     BETWEEN
663                     — THE MOUSELINE AND THE LINEPOSTOLOCV
```

Appendix A - Mailbox Displayer Code - Page 14

```
                        — THE MOUSELINE GIVES LINE WITHIN FIELD TOTAL
                        — THE LINEPOSTOLOCV USES LINE OF FIELD ON SCREEN
                        set mbxG_subtractLine = mbxG_subtractLine - 1
                        set numberOfIcons = the lineCount of member "MailboxTo" + 40
                        set amountOfMail = the lineCount of member "MailboxTo"
                        — put "linecount:" & the lineCount of member "MailboxTo"
                        — put "subtractline:" & mbxG_subtractLine
                        repeat with i = 40 to numberOfIcons
                                if mbxG_subtractLine >= - amountOfMail + 1 then
                                        set the locV of sprite i = the locV of sprite i - 15
                                        if the locV of sprite i <105 then
                                                set the visible of sprite i = FALSE
                                        else
                                                set the visible of sprite i = TRUE
                                        end if
                                else
                                        nothing
                                end if
                        end repeat

                        if mbxG_subtractLine < - amountOfMail + 1 then
                                set mbxG_subtractLine = - amountOfMail + 1
                        end if

                        — MOVE HIGHLIGHT WITH LINE, MOVING HIGHLIGHT
                        — OFF SCREEN WHEN LINE MOVES OFF SCREEN

                        set whichHighlight = mbxG_whichLine + mbxG_subtractLine
                        if whichHighlight <= 0 or whichHighlight >= 22 then
                                set the loc of sprite 11 to point (11, -20)
                        else
                                set the locV of Sprite 11 to (99 + linePosToLocV(member
                                "MailboxTo", whichHighlight))
                        end if
                        updateStage
                end repeat
        end if
end

—--

on emc_indicateCheckingInternet
        global mbxG_red
        global mbxG_blue
        global mbxG_white
        global mbxG_black

        if the locH of sprite 4 > 600 then
                set the loc of sprite 4 = point(223, 4)
        end if
        — if the locH of sprite 5 > 600 then
        —       set the loc of sprite 5 = point(509, 19)
        — end if
```

Appendix A - Mailbox Displayer Code - Page 15

```
717
718         set colorNow = the forecolor of line 1 of member the member of sprite 4
719
720         case colorNow of
721                 mbxG_black: set colorNext = mbxG_blue — blue
722                 mbxG_blue: set colorNext = mbxG_white — pink
723                 mbxG_white: set colorNext = mbxG_red — red
724                 mbxG_red: set colorNext = mbxG_blue — blue
725         end case
726
727         set the forecolor of line 1 of member the member of sprite 4 to colorNext
728         updateStage
729
730 end emc_indicateCheckingInternet
731
732 on emc_endIndicateCheckingInternet
733         set the loc of sprite 4 to point(800, 4)
734         set the loc of sprite 5 = point(800, 19)
735         cursor -1
736         updateStage
737 end emc_endIndicateCheckingInternet
738
739 ——— Scripts for the close window button
740
741 on mouseDown
742
743         repeat while the stillDown
744                 if inside(point(the mouseH, the mouseV), the rect of sprite the clickon) then
745                         if the name of member the member of sprite the clickon = "closeWindow"
746                         then
747                                 set the member of sprite the clickon = "closeWindow_down"
748                                 updateStage
749                         end if
750                 else
751                         set the member of sprite the clickon = "closeWindow"
752                         updateStage
753                 end if
754         end repeat
755         set the member of sprite the clickon = "closeWindow"
756         updateStage
757
758 end mouseDown
759
760 on mouseUp
761
762         — Close the window
763         if inside(point(the mouseH, the mouseV), the rect of sprite the clickon) then
764                 — these next to lines are to try and speed up
765                 — the disposal of the mailbox icons on close
766                 — need to check this on slower machine.
767                 hideMailIcons(the lineCount of member "MailboxTo")
```

Appendix A - Mailbox Displayer Code - Page 16

```
768                 go frame "stop"
769                 tell the stage to emh_killComponent(0, "")
770                 set success = the result
771                 if success <> TRUE then
772                         alert("error closing mailbox MIAW")
773                 end if
774         end if
775 end
776
777 on hideMailIcons numberOfIcons
778         repeat with i = 40 to (40 + numberOfIcons)
779                 set the visible of sprite i = FALSE
780         end repeat
781         updateStage
782 end
783
784
785 — script for the open button
786
787 on mouseDown
788
789         repeat while the stillDown
790                 if inside(point(the mouseH, the mouseV), the rect of sprite the clickon) then
791                         if the name of member the member of sprite the clickon = "open" then
792                                 set the member of sprite the clickon = "open_down"
793                                 updateStage
794                         end if
795                 else
796                         set the member of sprite the clickon = "open"
797                         updateStage
798                 end if
799         end repeat
800         set the member of sprite the clickon = "open"
801         updateStage
802 end mouseDown
803
804
```

Appendix A - Mailbox Displayer Code - Page 17

```
805  on mouseUp
806
807      if inside(point(the mouseH, the mouseV), the rect of sprite the clickon) then
808          global mbxG_whichLine, mbxG_messages
809          if mbxG_whichLine = 0 then
810              alert "Select a message by clicking with your mouse."
811              exit
812          end if
813          set maildata = getAt(mbxG_messages, mbxG_whichLine)
814          tell the stage
815              emh_openMessage(maildata)
816          end tell
817      end if
818  end
```

Appendix B - Type Updater Code - Page 1

```
--- TYPE_UPDATER IMPLEMENTATION
---- Code for a component that maintains message type information for the
---- KidCode electronic mail client.
---- This Director MIAW makes public functions available for calling by other components
of KidCode

--- public functions
--- 1. Initialize_TypeTable
---- 2. Install_Type
---- 3. Uninstall_Type



----- private functions fifor internal use only
----- 1. Write_Typetable_File
----- 2. Read_Typetable_File
---  3. Read_Icon_Files_To_RAM
----  4. read_iconFile
----  5. delete_mimetype
----  6. insert_mimetype
----  7. delete_filetype
----  8. insert_filetype


--- Filename for permanent storage version of TYPETABLE file
--- typetable.txt     --default directory is the currentPath directory


--- INITIALIZE_TYPETABLE initializes the data structures used to lookup Mime type
icons,  attachment filetype icons and message handler MIAWs.
--- The SG_Typetable is set up prior to itfs use.  File attachment information is not looked
up until it is used when a message with an attached file is encountered.

on Initialize_TypeTable
global SG_TYPETABLE        --- super global variable shared across different MIAWS for
Typetable
global SG_ATTACH_TYPETABLE    ---- maintains info for attachment filetypes

set SG_TYPETABLE = [:]    -- initialize property list for mimetype information

-- initialize property list for filetype information
-- this list will be filled only as messages with attachments are encountered
set SG_ATTACH_TYPETABLE = [:]
set SG_TYPETABLE = Read_TypeTable_File(the pathname & " typetable.txt")

if count(SG_TYPETABLE) = 0 then    -- failed to read typetable file
  alert ("Error: Failed to read the file of MIMEtypes")
  return (0)
end if

set retVal = Read_Icon_Files_To_RAM()
```

Appendix B - Type Updater Code - Page 2

```
52
53      if retVal = 0 then
54        alert("Error: Failed to load MIME type icons.)
55        return (0)
56      else
57        return(1)
58      end if
59
60      end   --- Initialize_TypeTable
61
62
63
64      --- INSTALL_TYPE is used to install a new MIME type into the system.
65      INSTALL_TYPE takes as input a mimetype (string), a filename of the message handling
66      movie, an filename of the bitmap that  contains the mimetype icon and, optionally, a file
67      extension (string).  The function adds the information associated with the MIME type
68      (given by the function parameters) into the MIME type table recorded in permanent storage.
69      Here we use the file "typetable.txt" for permanent storage of the MIME type info.
70
71      on Install_Type  mimeToInstall, msgHandler_filename, icon_filename, filetype
72      global SG_TYPETABLE        --- information on all installed MIME types
73
74      set DEFAULT_ICONFILE = "defaultIcon.bmp"
75      set SG_TYPETABLE  = [:]    -- initialize property list for mimetype information
76
77      ---read the existing MIME type information into RAM
78      set SG_TYPETABLE  = Read_TypeTable_File(the pathname & " typetable.txt")
79
80      if  count(SG_TYPETABLE) = 0 then    -- failed to read typetable file
81        alert ("Error: Failed to read the file of MIMEtypes")
82        return (0)
83      end if
84
85      ---Check to see if mimetype is already installed
86
87      set mimeProperties = get_mimetype(mimeToInstall)
88
89      if mimeProperties <> 0 then   ---mimetype is already installed
90        set redefineAlert = baMsgBox(theMessage, "KidCode", "YesNoCancel", "Question", 1)
91
92        -- the alert function should not save the message, only do the alert
93        case redefineAlert of
94              "No":  return 0
95              "Cancel": return fi1
96              otherwise:    nothing  -- continue
97        end case
98      end if
99
100     --- Define the new mimetype
101
102     if verifyMessageHandler(msgHandler_filename) = 0 then   --something wrong with
```

Appendix B - Type Updater Code - Page 3

```
103       program file
104         alert("Error: invalid message handler program" && msgHandler_filename)
105         return(0)
106       end if
107
108       insert_msgHandler(SG_TYPETABLE, mimeToInstall, msgHandler_filename)
109
110       if verifyIconImage(icon_filename) = 0 then  --something wrong with icon file
111         alert("Error: invalid icon file" && msgHandler_filename ". Using default icon.")
112         insert_iconFileName(SG_TYPETABLE, mimeToInstall, DEFAULT_ICONFILE)
113       else
114         insert_iconFileName(SG_TYPETABLE, mimeToInstall, icon_filename)
115       end if
116
117       if filetype <>"" then
118         insert_fileytype(mimeToInstall, filetype)
119         writeTypeToRegistry(mimeToInstall, filetype)
120       end if
121
122       set retVal = write_TypeTable_File()
123       if retVal = 0 then
124       alert("Error writing typetable to file." && mimeToInstall && "not installed.")
125       return(0)
126       else return(1)
127
128       end   --- Install_Type
129
130
131       ----- UNINSTALL_TYPE removes a mimetype and itís properties from both the file and
132       the global variable SG_TYPETABLE
133
134       on unInstall_Type  mimeType
135       global SG_TYPETABLE        --- information on all installed MIME types
136
137       set SG_TYPETABLE  = [:]    -- initialize property list for mimetype information
138
139       ---read the existing MIME type information into RAM
140       set SG_TYPETABLE  = Read_TypeTable_File(the pathname & " typetable.txt")
141
142       if  count(SG_TYPETABLE) = 0 then    -- failed to read typetable file
143         alert ("Error: Failed to read the file of MIMEtypes")
144         return (0)
145       end if
146
147       delete_mimetype(mimeType)
148
149       --- write the revised typetable to the file
150       set retVal = write_TypeTable_File()
151       if retVal = 0 then
152       alert("Error." && mimeToInstall && "could not be uninstalled. Typetable file write
153       error.")
```

Appendix B - Type Updater Code - Page 4

```
154     return(0)
155     else return(1)
156
157     end unInstall_Type
158
159     --- WRITE_TYPETABLE_FILE writes the information in SG_TYPETABLE to the
160     typetable file on disk.  This file stores properties associated with each mimetype.
161     ---- SG_TYPETABLE is a property list that contains a list of mimetypes.
162
163     on Write_Typetable_File
164     global SG_TYPETABLE        --- super global variable shared across different MIAWS
165     set fileName = the pathname & "typetable.txt"
166     set bkupFileName = the pathname & "typetable.bak"
167
168     if count( SG_TYPETABLE) = 0 then     -- no mimetypes defined
169           alert("Error. No mimetype data to write.)
170     return(0)
171     end if
172
173     --- create backup for typetable file
174     copyFile(filename, bkFileName)
175
176     -- start up Fileio Xtra
177      set mFile = new(xtra "fileio")
178
179      set retVal = deleteFile(mFile, filename)    -- delete old version before rewriting
180      set retVal = createFile(mFile, filename)
181      if retVal = 0 then
182           alert("Error updating typetable file.")
183           renameFile(bkFileName, filename)
184           return(0)
185      end if
186
187      openFile(mFile, fileName, 2)  -- open for write access
188      setPosition(mFile, 0)
189
190     --- write the data into the file
191     set i= 1
192     set mimeType = getAt(SG_TYPETABLE, i)
193     repeat while mimeType <> 0
194           set dataToWrite = mimetype
195           put " " & get_filetype(mimetype) into dataToWrite
196           put " " & get_iconFileName(mimetype) into dataToWrite
197     put " " & get_MsgHandler(mimetype) into dataToWrite
198     writeline(mFile, dataToWrite)
199     set i= i+1
200     set mimeType = getAt(SG_TYPETABLE, i)
201     end repeat
202
203     closeFile(mFile)
204     set retVal = deleteFile(mFile, bkFileName)    -- delete backup file
```

Appendix B - Type Updater Code - Page 5

```
205     return(0)
206     end   Write_TypeTable_File
207     --- READ_TYPETABLE_FILE reads the typetable file
208     ---- and creates a data structure in memory, SG_TYPETABLE
209     ---- SG_TYPETABLE is a property list that contains a list of mimetypes
210
211     When the function returns, the global property list data structure, SG_TYPETABLE,
212     contains an entry for each mimetype. Along with the pathName for the message handling
213     movie and the IconFile. Later the cast member number for the icon in RAM will be added
214     to the datastructure. For now these are all set to 0. This data structure looks like,
215     [ "text/plain": ["txt",  0, "C:\KidCode\text.gif", "C:\KidCode\text.dxr"], "x-
216     application/grid": ["",  0, "C:\KidCode\grid.gif", "C:\KidCode\grid.dxr"]]
217
218     on Read_TypeTable_File
219     global SG_TYPETABLE        --- super global variable shared across different MIAWS
220     set fileName = the pathname & "typetable.txt"
221
222     set SG_TYPETABLE  = [:]    -- initialize property list for mimetypes
223
224     -- start up FileIo Xtra
225      set mFile = new(xtra "fileio")
226      openFile(mFile, fileName, 1)  -- open for read only access
227      set status = status(mFile)
228
229      if status <> 0 then
230       alert("Error. Could not open mimetype table: " & error(mFile, status))
231       closeFile(mFile)  -- just to be safe
232       return FALSE
233      end if
234
235      setPosition(mFile, 0)
236
237      -- Lingo canft read one line at a time so simulate this by reading the entire file into the
238     string, str
239      set str = readFile(mFile)
240
241      set nTypes = the number of lines in str
242       repeat with j = 1 to nTypes
243            set mimetype = word 1 of line j
244            insert_mimetype(mimetype)
245            insert_filetype(mimetype, word 2 of line j)
246            insert_iconFileName(mimetype, word 3 of line j)
247            insert_msgHandler(mimetype, word 4 of line j)
248       end repeat
249
250      closeFile(mFile)
251
252     end   Read_TypeTable_File
253
254
255     on Read_Icon_Files_To_RAM
```

Appendix B - Type Updater Code - Page 6

```
256     global SG_TYPETABLE
257     global SG_DEFAULT_ICON_PTR =  1000
258
259     if count( SG_TYPETABLE) = 0 then     -- no mimetypes defined
260             alert("Error. No mimetype data. Canít load icons.)
261     return(0)
262     end if
263
264     ---load the default icon
265     importFileInto(member SG_DEFAULT_ICON_PTR, the pathname & "defaulticon.gif")
266
267     --- cycle through the mimetypes loading icons
268     set castNum = SG_DEFAULT_ICON_PTR + 1   --first icon immediately follows the
269     default
270     set i= 1
271     set mimeType = getAt(SG_TYPETABLE, i)
272     repeat while mimeType <> 0
273             set iconFile = get_iconFileName(SG_TYPETABLE, mimetype)
274             if icon ="" then  --- icon not defined use default
275                     set iconPtr = SG_DEFAULT_ICON_PTR
276             else
277     set iconPtr = read_IconFile(iconFile, castNum )
278     if iconPtr  > 0 then
279             set castNum = castNum + 1
280     else set iconPtr = SG_DEFAULT_ICON_PTR
281             end if
282
283             insert_iconPtr(mimetype, iconPtr)
284     set i= i+1
285     set mimeType = getAt(SG_TYPETABLE, i)
286     end repeat
287
288     return(1)
289
290     end Read_Icon_Files_To_RAM
291
292
293
294     --- READ_ICONFILE loads a single icon bitmap into RAM
295     on read_IconFile filename, castMemberNum
296     set retVal = importFileInto(member castNum, iconFile)
297     return (retVal)
298     end read_IconFile
299     --- Data Access Functions for MIMETYPE info
300     --- Data is stored in the property list SG_TYPETABLE with the following structure:
301     --- mimetype: properties
302     --- where mimetype is a string, e.g. "text/plain"
303     --- and properties is a list with the following elements:
304     ----      [ filetype, iconPtr, iconFilename, msgHandler_FileName
305     ----  e.g. SG_TYPETABLE =
306     ----      [ "text/plain": ["txt",  0, "C:\KidCode\text.gif", "C:\KidCode\text.dxr"],
```

Appendix B - Type Updater Code - Page 7

```
307     ----        "x-application/grid": ["", 0, "C:\KidCode\grid.gif", "C:\KidCode\grid.dxr"]]
308
309     on get_mimetype mimetype
310     global SG_TYPETABLE
311     return( getProp(SG_TYPETABLE, mimetype) )
312     end
313
314     on get_filetype mimetype
315     global SG_TYPETABLE
316     set theProperties = getProp(SG_TYPETABLE, mimetype)
317     return( getAt(theProperties, 1) )
318     end
319
320     on get_iconPtr mimetype
321     global SG_TYPETABLE
322     set theProperties = getProp(SG_TYPETABLE, mimetype)
323     return( getAt(theProperties, 2) )
324     end
325
326     on get_iconFileName mimetype
327     global SG_TYPETABLE
328     set theProperties = getProp(SG_TYPETABLE, mimetype)
329     return( getAt(theProperties, 3) )
330     end
331
332     on get_msgHandler mimetype
333     global SG_TYPETABLE
334     set theProperties = getProp(SG_TYPETABLE, mimetype)
335     return( getAt(theProperties, 4) )
336     end
337
338     on insert_mimetype mimetype
339     global SG_TYPETABLE
340     addProp(SG_TYPETABLE, mimetype)
341     end
342
343     on insert_filetype mimetype, filetype
344     global SG_TYPETABLE
345     set theProperties = getProp(SG_TYPETABLE, mimetype)
346             add(theProperties, filetype)
347             setProp(SG_TYPETABLE, mimetype, theProperties)
348     end
349
350     on insert_iconPtr mimetype, iconPtr
351     global SG_TYPETABLE
352     set theProperties = getProp(SG_TYPETABLE, mimetype)
353             add(theProperties, iconPtr)
354             setProp(SG_TYPETABLE, mimetype, theProperties)
355     end
356
357     on insert_iconFileName mimetype, iconFilename
```

Appendix B - Type Updater Code - Page 8

```
358     global SG_TYPETABLE
359     set theProperties = getProp(SG_TYPETABLE, mimetype)
360             add(theProperties, iconFilename)
361             setProp(SG_TYPETABLE, mimetype, theProperties)
362     end
363
364     on insert_msgHandler  mimetype
365     global SG_TYPETABLE
366     set theProperties = getProp(SG_TYPETABLE, mimetype)
367             add(theProperties, msgHandler)
368             setProp(SG_TYPETABLE, mimetype, theProperties)
369     end
370
371
372     on delete_mimetype  mimetype
373     global SG_TYPETABLE
374     deleteProp(SG_TYPETABLE, mimetype)
375     end
376
377     on delete_filetype  mimetype
378     global SG_TYPETABLE
379     set properties = getProp(SG_TYPETABLE, mimetype)
380             setAt(properties, 1,"")
381             setProp(SG_TYPETABLE, mimetype, properties)
382     end
383
384     on delete_icon  mimetype
385     global SG_TYPETABLE
386     set properties = getProp(SG_TYPETABLE, mimetype)
387             setAt(properties, 2, 0)
388             setProp(SG_TYPETABLE, mimetype, properties)
389     end
390
391     on delete_iconFileName  mimetype
392     global SG_TYPETABLE
393     set properties = getProp(SG_TYPETABLE, mimetype)
394             setAt(properties, 3,"")
395             setProp(SG_TYPETABLE, mimetype, properties)
396     end
397
398     on delete_msgHandler  mimetype
399     global SG_TYPETABLE
400     set properties = getProp(SG_TYPETABLE, mimetype)
401             setAt(properties, 4,"")
402             setProp(SG_TYPETABLE, mimetype, properties)
403     end
```

Claims:

1. Electronic mail client software for use with a display device, comprising:
a) mailbox displayer means for displaying the contents of a mailbox as a scrollable list on the display device wherein each item of mail is listed with a plurality of properties, said properties selected from the group consisting of subject, sender's name, and date sent;
b) association means for associating a plurality of mime types with a plurality of icon images, wherein
said mailbox displayer means includes means for determining the mime type of at least some items of mail in the mailbox, means for reading said association means, and means for displaying in the scrollable list an icon image as a property associated with each of said at least some items of mail, said icon image for each of said at least some items of mail being selected from said association means according to the mime type for each of said at least some items of mail.

2. Electronic mail client software according to claim 1, further comprising:
c) type updater means for updating said association means to include additional mime types and additional icon images.

3. Electronic mail client software according to claim 1, further comprising:
c) a plurality of icon images, each having a filename, wherein
said association means is a data structure associating each of at least some mime types with the filenames of an icon image.

4. Electronic mail client software according to claim 1, wherein:
said association means includes means for associating mime types with programs, and
wherein some mimetypes are not associated with icon images but are associated with programs.

5. Electronic mail client software according to claim 2, wherein:
said type updater means is responsive to user input.

6. Electronic mail client software according to claim 3, wherein:
at least some of said plurality of images are scalable.

7. Electronic mail client software according to claim 1, wherein:

said electronic mail client software is designed to be used with an operating system which maintains a registry of icons, and

said association means associates at least some mime types with icons selected from said registry of icons.

8. Electronic mail client software according to claim 1, further comprising:

c) icon recovery means for reading graphical icon information contained in an email or email attachment, wherein

said association means associates a mime type with an icon recovered by said icon recovery means if no other icon is found.

9. Electronic mail client software according to claim 2, wherein:

said type updater means is manually operable by a user of said electronic mail client software.

10. Electronic mail client software according to claim 2, wherein:

said type updater means automatically installs a new icon image according to an event selected from the group consisting of

a new application or component is installed which is capable of authoring/reading a new mime type,

a new mime type is encountered in a received mail message, and

a regularly scheduled event causes said type updater to check a server for new icons.

11. A method of displaying a list of the contents of an electronic mail box on a display device, comprising:

a) displaying the contents of the mailbox as a scrollable list on the display device wherein each item of mail is listed with a plurality of properties, said properties selected from the group consisting of subject, sender's name, and date sent;

b) determining the mime type of at least some items of mail in the mailbox;

c) displaying in the scrollable list an icon image as a property associated with each of said at least some items of mail, said icon image for each of said at least some items of mail being selected according to the mime type for each of said at least some items of mail.

12. A method according to claim 11, further comprising:
d) associating mime types with programs;
e) executing a program associated with a mime type when a mail item of the mime type is opened.

13. A method according to claim 11, wherein:
at least some of said icon images are scalable.

14. A method according to claim 11, wherein:
said step of displaying includes selecting at least some icons from a central registry of icons.

15. A method according to claim 11, wherein:
said step of displaying includes reading graphical icon information contained in an email or email attachment.

16. A method according to claim 11, further comprising:
d) maintaining a store of graphical icons for use when performing said step of displaying.

17. A method according to claim 16, further comprising:
e) automatically installing a new icon image in said store of graphical icons in response to an event selected from the group consisting of
when a new application or component is installed which is capable of authoring/reading a new mime type,
when a new mime type is encountered in a received mail message, and
when a regularly scheduled event causes said type updater to check a server for new icons.

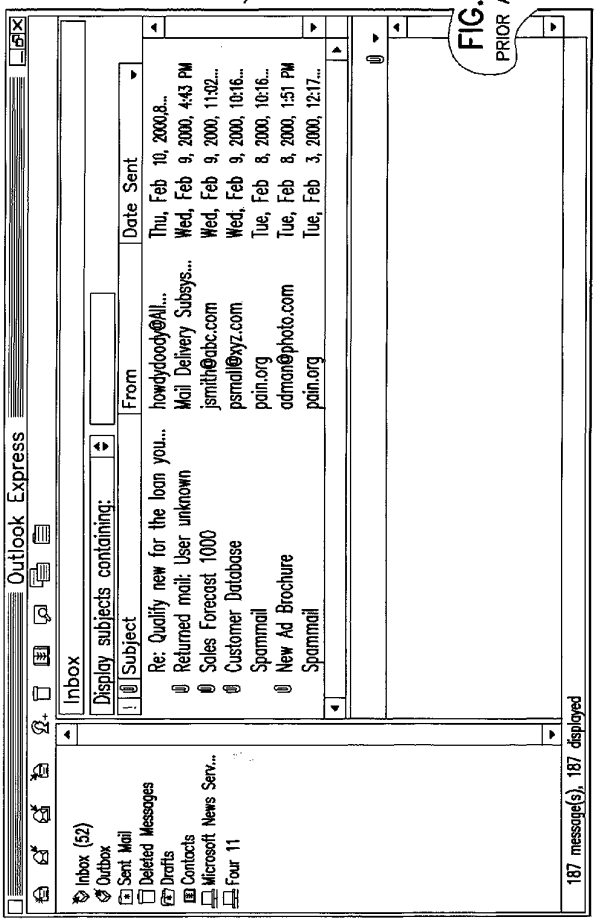


FIG.1
PRIOR ART

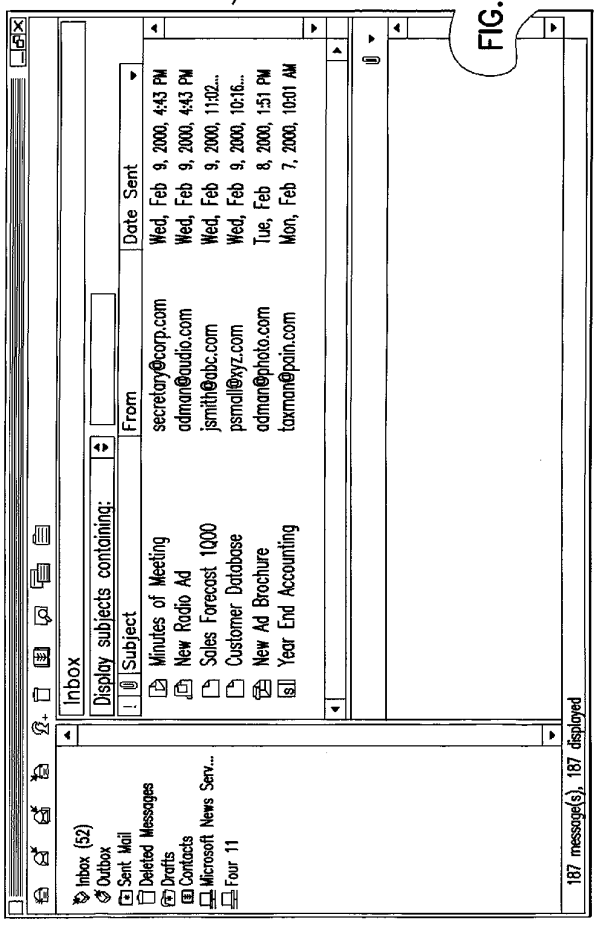

Inbox
Display subjects containing:
Inbox (52)
Outbox
Sent Mail
Deleted Messages
Drafts
Contacts
Microsoft News Serv...
Four 11
! | Subject | From | Date Sent
Minutes of Meeting secretary@corp.com Wed, Feb 9, 2000, 4:43 PM
New Radio Ad adman@audio.com Wed, Feb 9, 2000, 4:43 PM
Sales Forecast 1Q00 jsmith@abc.com Wed, Feb 9, 2000, 11:02...
Customer Database psmall@xyz.com Wed, Feb 9, 2000, 10:16...
New Ad Brochure admen@photo.com Tue, Feb 8, 2000, 1:51 PM
Year End Accounting taxman@pain.com Mon, Feb 7, 2000, 10:01 AM
187 message(s), 187 displayed

FIG.2

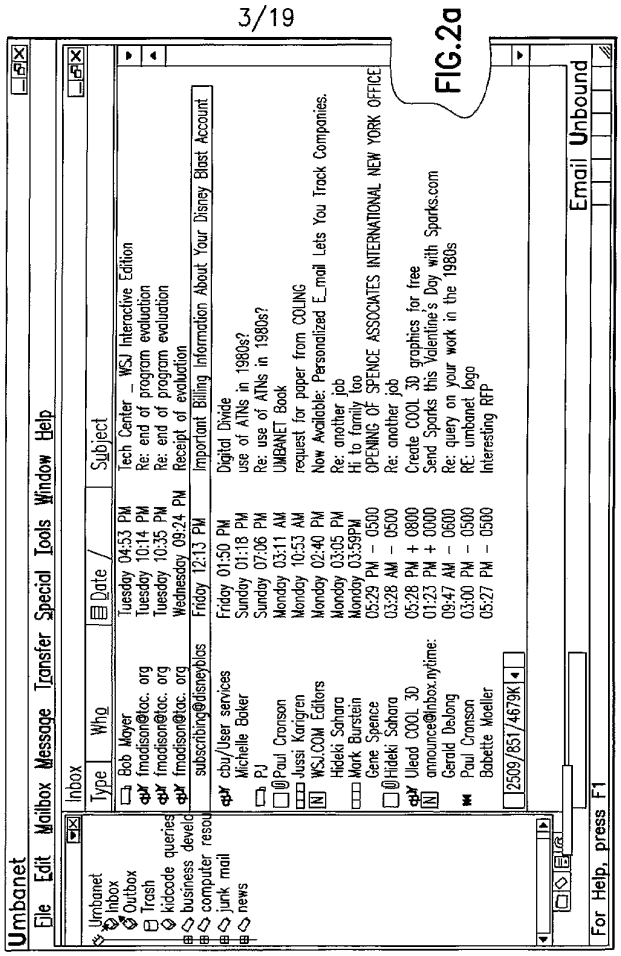

Umbanet
File Edit Mailbox Message Transfer Special Tools Window Help
Inbox
Type Who Date / Subject
Umbanet
Inbox
Outbox
Trash
kidcode queries
business develo
computer resou
junk mail
news
Bob Moyer Tuesday 04:53 PM Tech Center – WSJ Interactive Edition
fmadison@tac. org Tuesday 10:14 PM Re: end of program evaluation
fmadison@tac. org Tuesday 10:35 PM Re: end of program evaluation
fmadison@tac. org Wednesday 09:24 PM Receipt of evaluation
subscribing@disneyblas Friday 12:13 PM Important Billing Information About Your Disney Blast Account
cbu/User services Friday 01:50 PM Digital Divide
Michelle Baker Sunday 01:18 PM use of ATMs in 1980s?
PJ Sunday 07:06 PM Re: use of ATMs in 1980s?
Paul Cronson Monday 03:11 AM UMBANET Book
Jussi Karlgren Monday 10:53 AM request for paper from COLING
WSJ.COM Editors Monday 02:40 PM Now Available: Personalized E_mail Lets You Track Companies.
Hideki Sahara Monday 03:05 PM Re: another job
Mark Burstein Monday 03:59PM Hi to family too
Gene Spence 05:29 PM – 0500 OPENING OF SPENCE ASSOCIATES INTERNATIONAL NEW YORK OFFICE
Hideki Sahara 03:28 AM – 0500 Re: another job
Ulead COOL 3D 05:28 PM + 0800 Create COOL 3D graphics for free
announce@inbox.nytime: 01:23 PM + 0000 Send Sparks this Valentine's Day with Sparks.com
Gerald DeJong 09:47 AM – 0600 Re: query on your work in the 1980s
Paul Cronson 03:00 PM – 0500 RE: umbanet logo
Babette Moeller 05:27 PM – 0500 Interesting RFP
2509/851/4679K
For Help, press F1
Email Unbound

FIG.2a

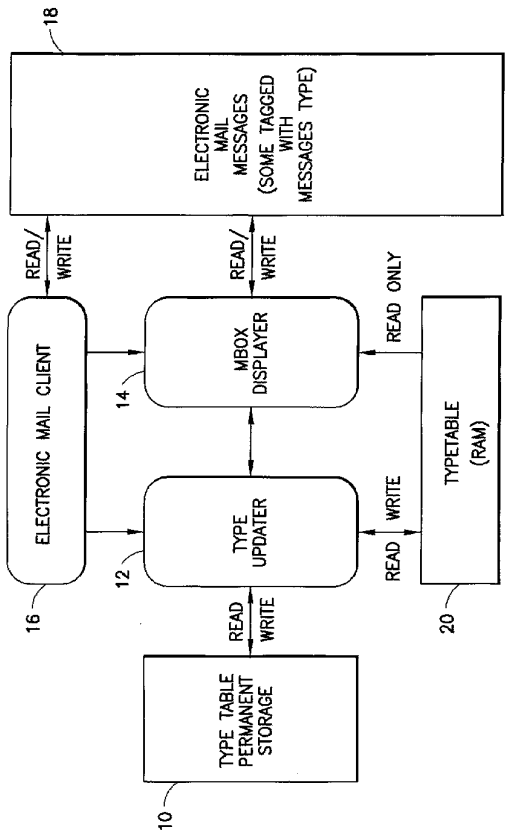

18
ELECTRONIC MAIL MESSAGES (SOME TAGGED WITH MESSAGES TYPE)
READ/ WRITE
READ/ WRITE
READ ONLY
ELECTRONIC MAIL CLIENT
MBOX DISPLAYER
14
TYPE UPDATER
12
WRITE
READ
TYPETABLE (RAM)
16
20
READ
WRITE
TYPE TABLE PERMANENT STORAGE
10

FIG.3

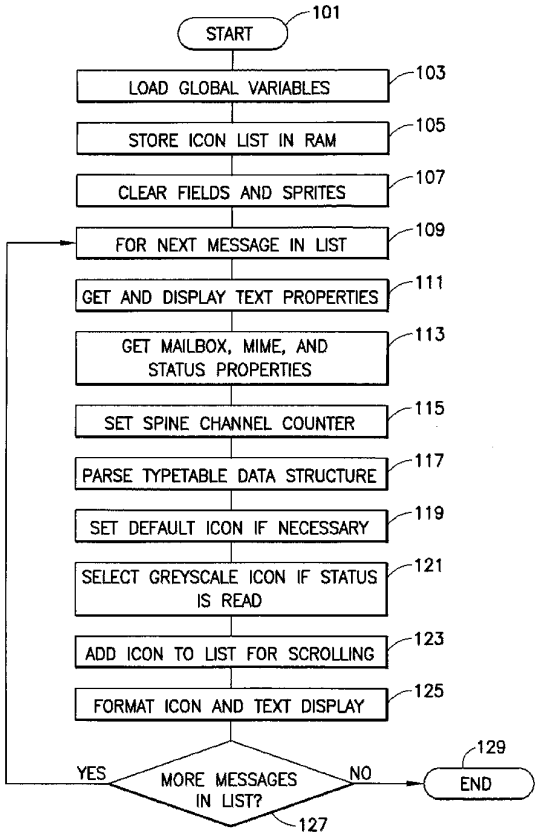

START
101
LOAD GLOBAL VARIABLES
103
STORE ICON LIST IN RAM
105
CLEAR FIELDS AND SPRITES
107
FOR NEXT MESSAGE IN LIST
109
GET AND DISPLAY TEXT PROPERTIES
111
GET MAILBOX, MIME, AND STATUS PROPERTIES
113
SET SPINE CHANNEL COUNTER
115
PARSE TYPETABLE DATA STRUCTURE
117
SET DEFAULT ICON IF NECESSARY
119
SELECT GREYSCALE ICON IF STATUS IS READ
121
ADD ICON TO LIST FOR SCROLLING
123
FORMAT ICON AND TEXT DISPLAY
125
MORE MESSAGES IN LIST?
127
YES
NO
END
129

FIG.4

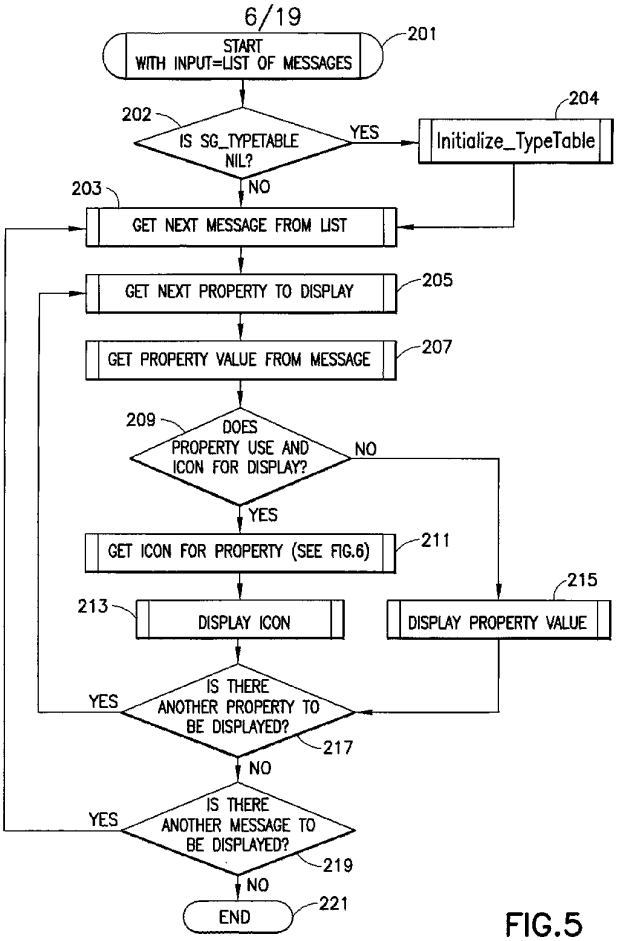

START WITH INPUT=LIST OF MESSAGES
201
IS SG_TYPETABLE NIL?
202
YES
Initialize_TypeTable
204
NO
GET NEXT MESSAGE FROM LIST
203
GET NEXT PROPERTY TO DISPLAY
205
GET PROPERTY VALUE FROM MESSAGE
207
DOES PROPERTY USE AND ICON FOR DISPLAY?
209
NO
YES
GET ICON FOR PROPERTY (SEE FIG.6)
211
DISPLAY ICON
213
DISPLAY PROPERTY VALUE
215
IS THERE ANOTHER PROPERTY TO BE DISPLAYED?
YES
217
NO
IS THERE ANOTHER MESSAGE TO BE DISPLAYED?
YES
219
NO
END
221

FIG.5

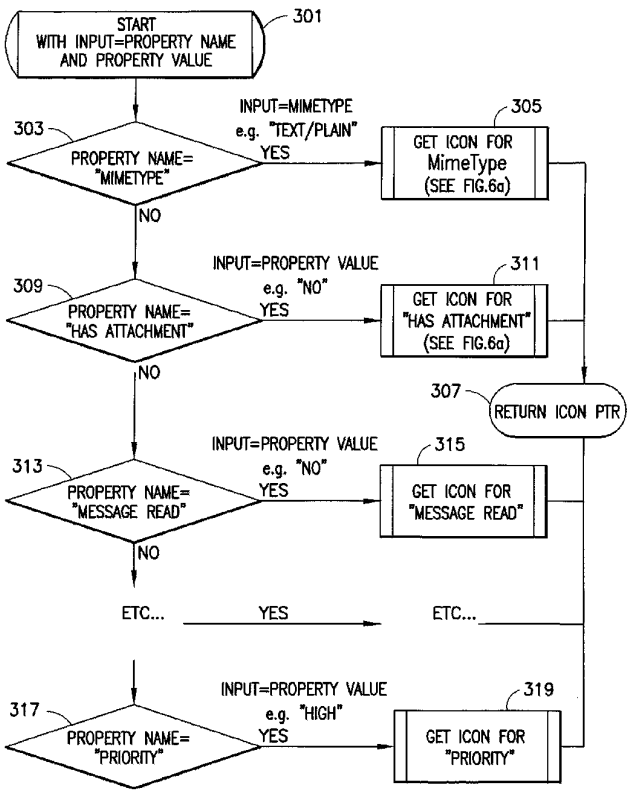

START
WITH INPUT=PROPERTY NAME
AND PROPERTY VALUE
301
303
PROPERTY NAME=
"MIMETYPE"
INPUT=MIMETYPE
e.g. "TEXT/PLAIN"
YES
305
GET ICON FOR
MimeType
(SEE FIG.6a)
NO
309
PROPERTY NAME=
"HAS ATTACHMENT"
INPUT=PROPERTY VALUE
e.g. "NO"
YES
311
GET ICON FOR
"HAS ATTACHMENT"
(SEE FIG.6a)
NO
307
RETURN ICON PTR
313
PROPERTY NAME=
"MESSAGE READ"
INPUT=PROPERTY VALUE
e.g. "NO"
YES
315
GET ICON FOR
"MESSAGE READ"
NO
ETC...
YES
ETC...
317
PROPERTY NAME=
"PRIORITY"
INPUT=PROPERTY VALUE
e.g. "HIGH"
YES
319
GET ICON FOR
"PRIORITY"

FIG.6

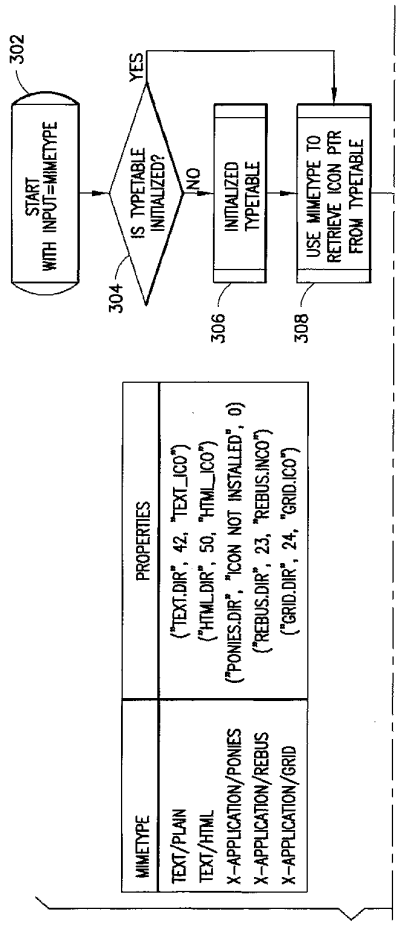


FIG.6a

FIG.6a-1

FIG.6a-2

FIG.6a-1

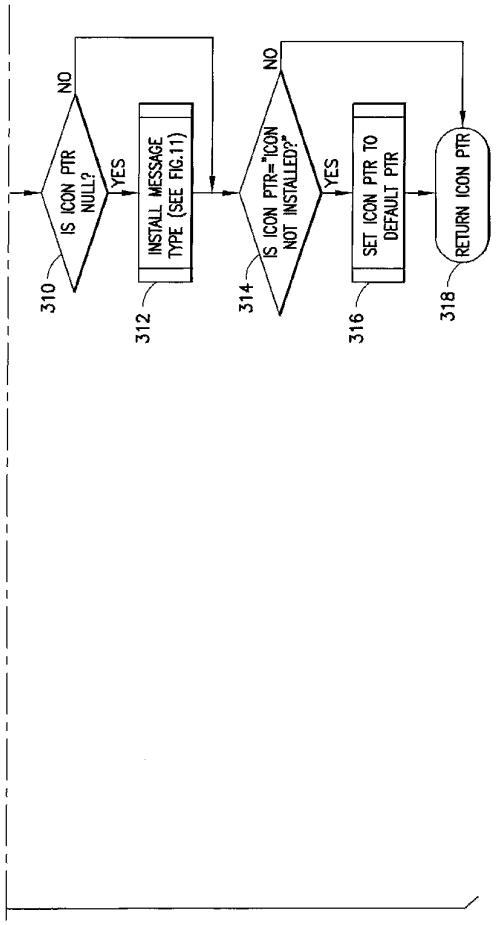

NO
IS ICON PTR NULL?
YES
INSTALL MESSAGE TYPE (SEE FIG.11)
NO
IS ICON PTR="ICON NOT INSTALLED?"
YES
SET ICON PTR TO DEFAULT PTR
RETURN ICON PTR
310
312
314
316
318

FIG.6a-2

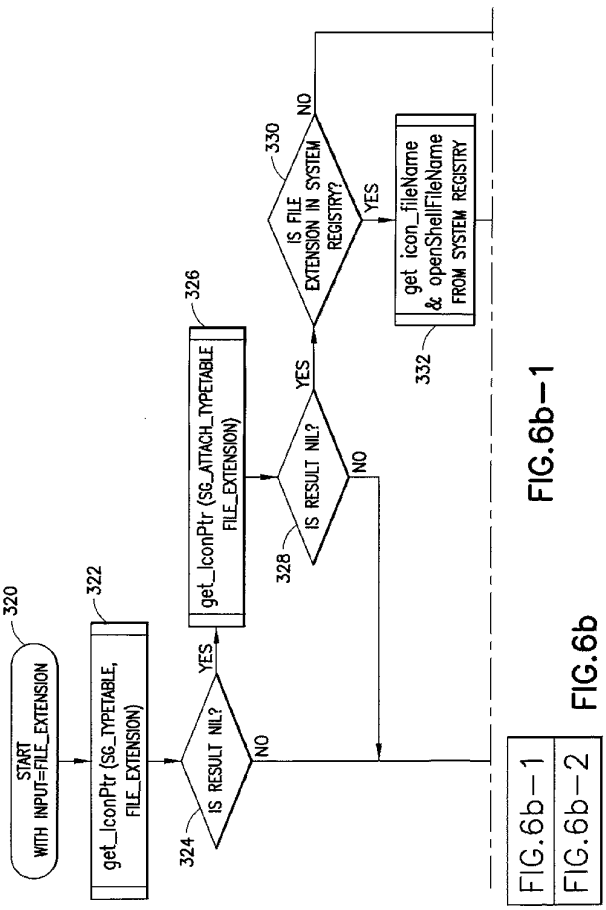

320
START
WITH INPUT=FILE_EXTENSION
322
get_IconPtr (SG_TYPETABLE, FILE_EXTENSION)
324
IS RESULT NIL?
YES
NO
326
get_IconPtr (SG_ATTACH_TYPETABLE FILE_EXTENSION)
328
IS RESULT NIL?
YES
NO
330
IS FILE EXTENSION IN SYSTEM REGISTRY?
YES
NO
332
get icon_fileName & openShellFileName FROM SYSTEM REGISTRY
FIG.6b-1
FIG.6b-2

FIG.6b

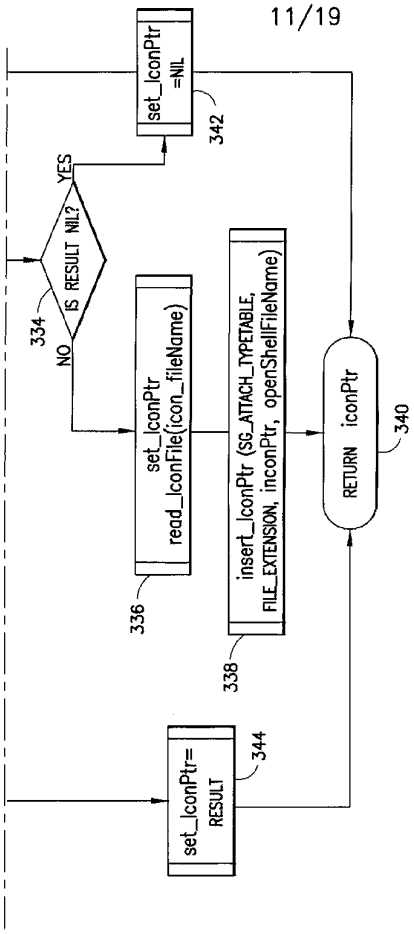

set_IconPtr =NIL
342
YES
IS RESULT NIL?
334
NO
set_IconPtr read_IconFile(icon_fileName)
336
insert_IconPtr (SG_ATTACH_TYPETABLE, FILE_EXTENSION, inconPtr, openShellFileName)
338
RETURN iconPtr
340
set_IconPtr= RESULT
344

FIG.6b-2

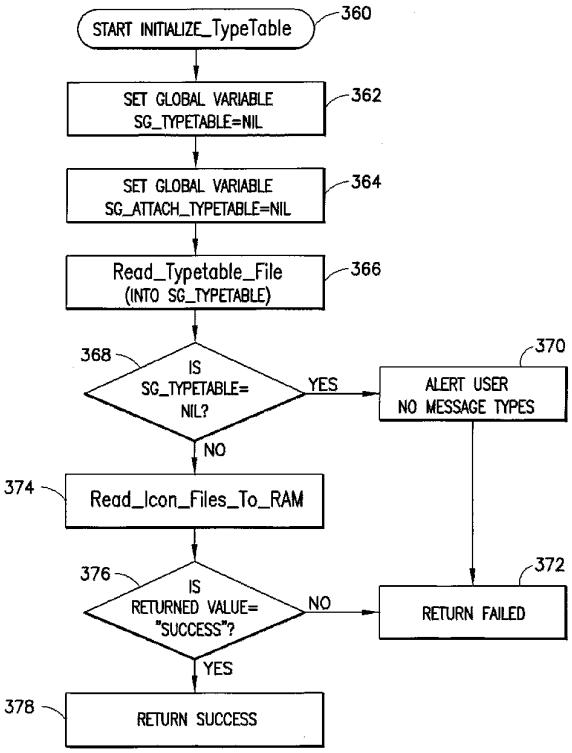

START INITIALIZE_TypeTable
360
SET GLOBAL VARIABLE SG_TYPETABLE=NIL
362
SET GLOBAL VARIABLE SG_ATTACH_TYPETABLE=NIL
364
Read_Typetable_File (INTO SG_TYPETABLE)
366
368
IS SG_TYPETABLE= NIL?
YES
370
ALERT USER NO MESSAGE TYPES
NO
374
Read_Icon_Files_To_RAM
376
IS RETURNED VALUE= "SUCCESS"?
NO
372
RETURN FAILED
YES
378
RETURN SUCCESS

FIG.7

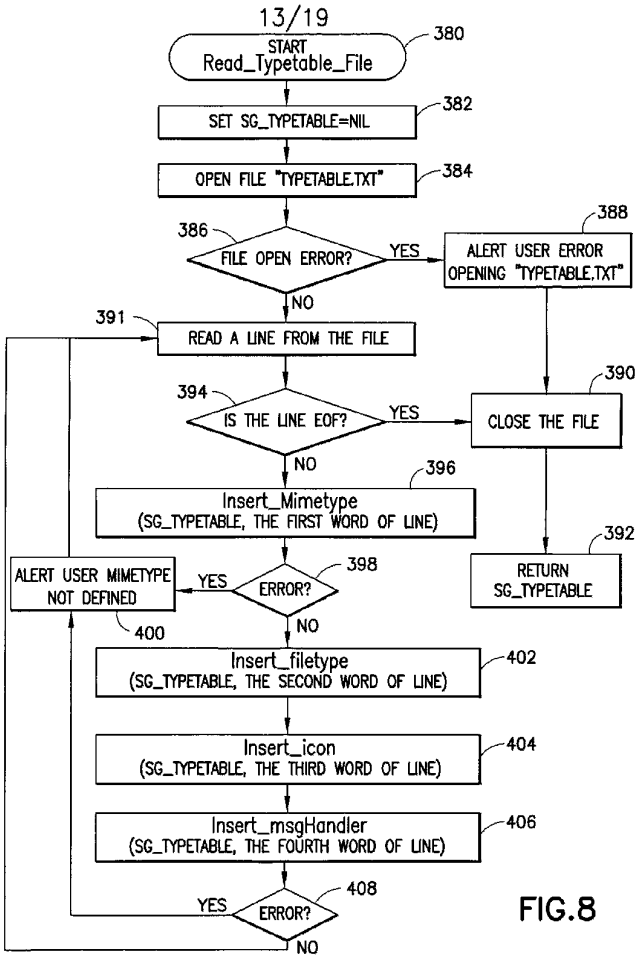

START
Read_Typetable_File
380
SET SG_TYPETABLE=NIL
382
OPEN FILE "TYPETABLE.TXT"
384
386
FILE OPEN ERROR?
YES
388
ALERT USER ERROR
OPENING "TYPETABLE.TXT"
NO
391
READ A LINE FROM THE FILE
394
IS THE LINE EOF?
YES
390
CLOSE THE FILE
NO
396
Insert_Mimetype
(SG_TYPETABLE, THE FIRST WORD OF LINE)
392
RETURN
SG_TYPETABLE
ALERT USER MIMETYPE
NOT DEFINED
YES
ERROR?
398
400
NO
Insert_filetype
(SG_TYPETABLE, THE SECOND WORD OF LINE)
402
Insert_icon
(SG_TYPETABLE, THE THIRD WORD OF LINE)
404
Insert_msgHandler
(SG_TYPETABLE, THE FOURTH WORD OF LINE)
406
YES
ERROR?
408
NO

FIG.8

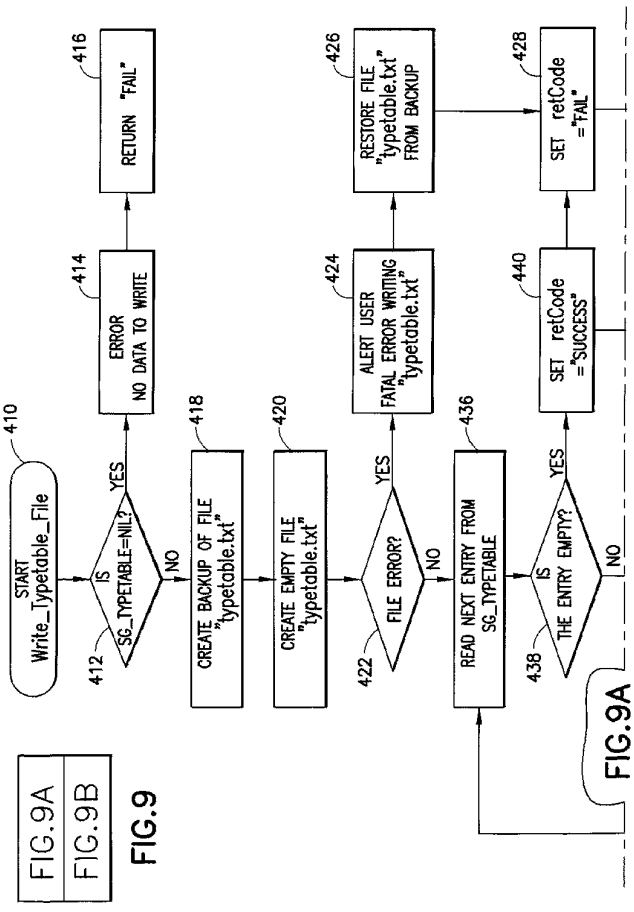


FIG.9A
FIG.9B

FIG.9

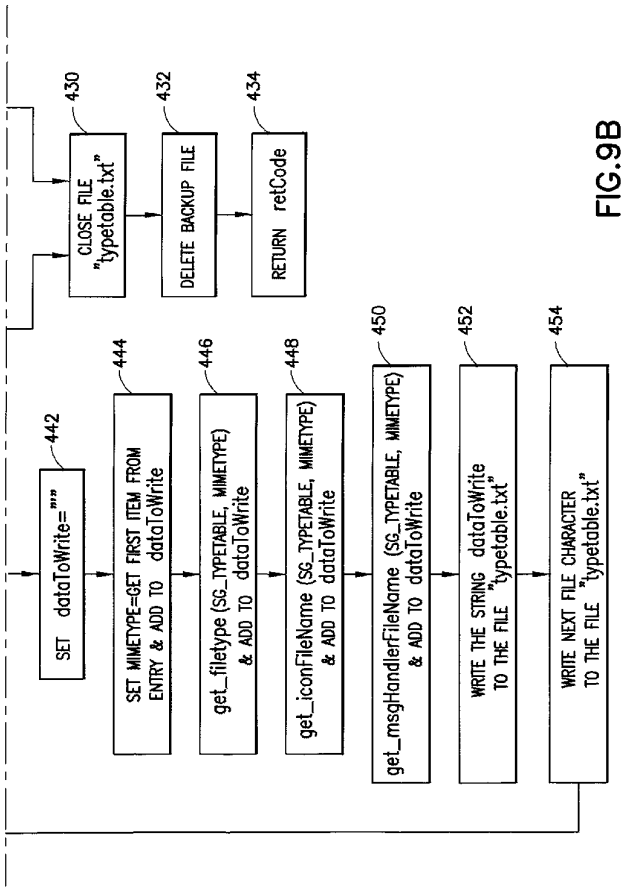

430
CLOSE FILE "typetable.txt"
432
DELETE BACKUP FILE
434
RETURN retCode
442
SET dataToWrite=""
444
SET MIMETYPE=GET FIRST ITEM FROM ENTRY & ADD TO dataToWrite
446
get_filetype (SG_TYPETABLE, MIMETYPE) & ADD TO dataToWrite
448
get_iconFileName (SG_TYPETABLE, MIMETYPE) & ADD TO dataToWrite
450
get_msgHandlerFileName (SG_TYPETABLE, MIMETYPE) & ADD TO dataToWrite
452
WRITE THE STRING dataToWrite TO THE FILE "typetable.txt"
454
WRITE NEXT FILE CHARACTER TO THE FILE "typetable.txt"

FIG.9B

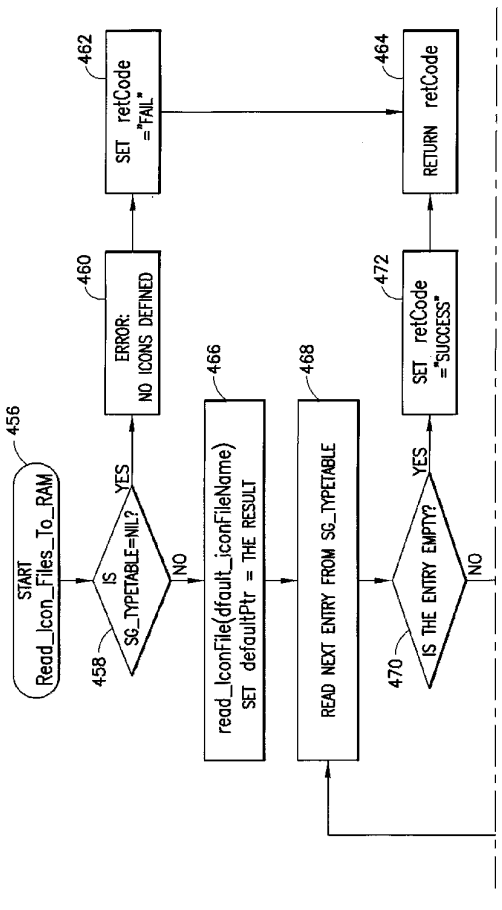


FIG.10A

FIG.10

FIG.10A
FIG.10B

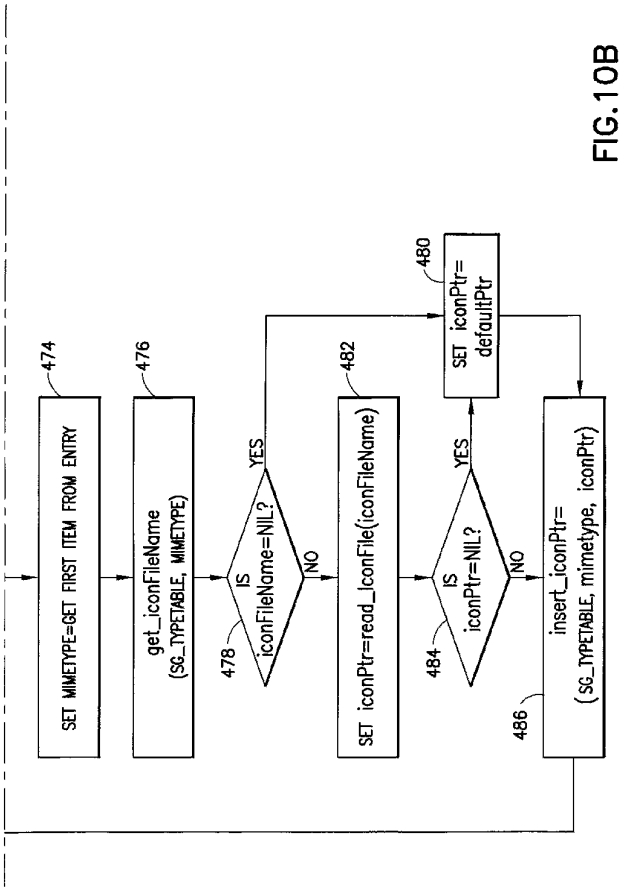

474
SET MIMETYPE=GET FIRST ITEM FROM ENTRY
476
get_iconFileName (SG_TYPETABLE, MIMETYPE)
478
IS iconFileName=NIL?
YES
NO
480
SET iconPtr= defaultPtr
482
SET iconPtr=read_IconFile(iconFileName)
484
IS iconPtr=NIL?
YES
NO
486
insert_iconPtr= (SG_TYPETABLE, mimetype, iconPtr)

FIG.10B

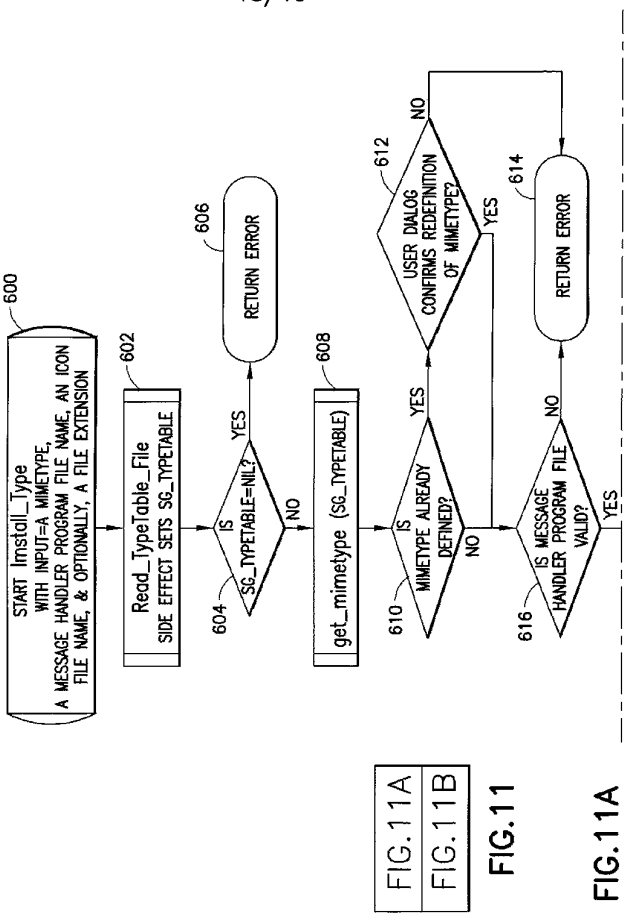


FIG.11A
FIG.11B

FIG.11

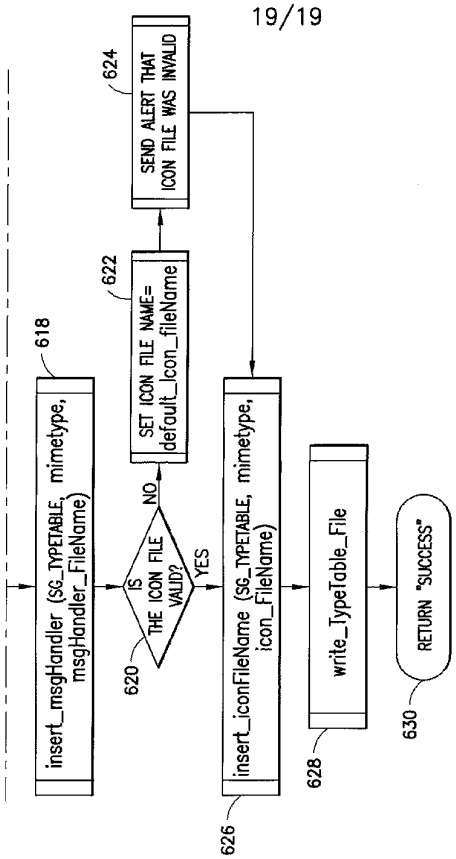

618
insert_msgHandler (SG_TYPETABLE, mimetype, msgHandler_FileName)
620
IS THE ICON FILE VALID?
NO
YES
622
SET ICON FILE NAME= default_Icon_fileName
624
SEND ALERT THAT ICON FILE WAS INVALID
626
insert_iconFileName (SG_TYPETABLE, mimetype, icon_FileName)
628
write_TypeTable_File
630
RETURN "SUCCESS"

FIG.11B

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/20348

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 13/00, 15/16.
US CL :709/206; 345/700.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/206; 345/700

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
west

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,E	US 6,092,114 A (SHAFFER et al) 18 July 2000	1-17
A,E	US 6,189,026 B1 (BIRRELL et al) 13 February 2001	1-17
A,P	US 6,073,166 A (FORSEN) 06 June 2000	1-17
A	US 5,752,059 A (HOLLERAN et al) 12 May 1998	1-17
A	US 5,734,901 A (SIDHU et al) 31 March 1998	1-17
A	US 5,826,062 A (FAKE, JR. et al) 20 October 1998	1-17

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
10 AUGUST 2001	29 AUG 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer GLENTON BURGESS
Facsimile No. (703) 305-3230	Telephone No. (703) 305-4792

Form PCT/ISA/210 (second sheet) (July 1998)*

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GW,ML,MR,NE,SN,TD,TG),AE,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,CA,CH,CN,CO,CU,CZ,DE,DK,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MD,MG,MK,MN,MW,MX,NO,NZ,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TR,TT,UA,UG,UZ,VN,YU,ZA,ZW

(72)発明者 ベイカー, ミシェル
アメリカ合衆国, ニューヨーク １００２５, ニューヨーク, リバーサイド ドライブ ３２５

【要約の続き】
が使用する拡大縮小可能なアイコンのファイル名と共に、メッセージタイプ及びサブタイプのリストが含まれている。