Oracle Email

アプリケーション開発者ガイド

リリース 2（9.0.4）

2003 年 11 月

部品番号 : J07726-02

ORACLE®

Oracle Email アプリケーション開発者ガイド , リリース 2（9.0.4）

部品番号 : J07726-02

原本名 ：Oracle Email Application Developer’s Guide, Release 2 (9.0.4)

原本部品番号 ：B10721-01

原本著者 ：Ginger Tabora

原本協力者 ：Vicky Cao, Ashish Consul, Vikas Dhamija, Tanya Hitaisinee, Harvinder Walia, Anthony Ye

目次

2 Java API リファレンス

3 SMTP スキャナ・インタフェース API

索引

はじめに

対象読者

『Oracle Email アプリケーション開発者ガイド』は、アプリケーション設計者および開発者を対象としています。PL/SQL でのプログラミング経験があることを前提としています。

このマニュアルの構成

第 1 章「PL/SQL API リファレンス」
この章では、Oracle Email システムで電子メールへのアクセスおよび管理に使用できる、一連の Oracle Email PL/SQL API について説明します。

第 2 章「Java API リファレンス」
この章では、カスタマイズされたクライアントの作成、アプリケーションの統合、特定拡張機能のサポートに使用できる Oracle Email Java API について説明します。

関連ドキュメント

詳細は、次の Oracle ドキュメントを参照してください。

- 『Oracle Email 管理者ガイド』
- 『Oracle Email Migration Tool Guide』
- 「Oracle Email API Reference」（Java Doc）
- 『Oracle Voicemail & Fax 管理者ガイド』

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

本文の表記規則

本文では、特定の項目が一目でわかるように、次の表記規則を使用します。次の表に、その規則と使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語および用語集に記載されている用語を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システム指定の要素を示します。このような要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドがあります。また、システム指定の列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみ、この句を指定できます。 BACKUP コマンドを使用して、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびユーザーが指定する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子があります。また、ユーザーが指定するデータベース・オブジェクトとデータベース構造、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。 **注意：** プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは、orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを作成します。 hr.departments 表には、department_id、department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスが次のメソッドを実装します。
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。ここで、*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文の例です。次のように固定幅フォントで表示され、通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用される表記規則とその使用例を示します。

規則	意味	例
[ ]	大カッコは、カッコ内の項目を任意に選択することを表します。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、カッコ内の項目のうち、1 つが必須であることを表します。中カッコは、入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択項目の区切りに使用します。項目のうちの 1 つを入力します。縦線は、入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関連しないコードの一部が省略されている。	CREATE TABLE ... AS *subquery*;
	■ コードの一部を繰り返すことができる。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の記号	大カッコ、中カッコ、縦線および省略記号以外の記号は、記載されているとおりに入力する必要があります。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システム指定の要素を示します。これらの要素は、ユーザー定義の要素と区別するために大文字で示されます。大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、大 / 小文字が区別されないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名などです。	`SELECT last_name, employee_id FROM employees;` `sqlplus hr/hr`
	注意： プログラム要素には、大文字と小文字を組み合せて使用するものもあります。これらの要素は、記載されているとおりに入力してください。	`CREATE USER mjones IDENTIFIED BY ty3MU9;`

1

PL/SQL API リファレンス

Oracle Email PL/SQL API は、Oracle Email システムで電子メールへのアクセスおよび管理に使用する一連の Application Program Interface です。

次の項目について説明します。

概要

この章で説明する PL/SQL API は、ビジネス要件やアプリケーション要件に合せてカスタマイズできる Oracle Email のシステム機能を公開しています。これらは、次の PL/SQL パッケージにまとめられています。

- MAIL_SESSION
- MAIL_FOLDER
- MAIL_MESSAGE
- MAIL_RECOVERY

Oracle Collaboration Suite をインストールすると、すべてのパッケージがインストールされます。最初の 3 つのパッケージは、電子メール・ユーザーのメール・アカウントを操作するためのパブリック・パッケージです。MAIL_RECOVERY パッケージは、ユーザーが誤って削除した電子メールをリカバリするための管理者用の機能です。

MAIL_SESSION パッケージ

MAIL_SESSION パッケージは、認証、ログアウト、およびクォータや使用情報の取得などのユーザー・セッション関連機能を提供します。

MAIL_FOLDER パッケージ

メール・フォルダは、電子メール・メッセージの整理に使用します。電子メール・ユーザーの作成中、すべての受信メッセージを保存する INBOX フォルダがデフォルトで作成されます。ユーザーはメール・フォルダを使用して、INBOX フォルダに入ったメッセージを整理できます。MAIL_FOLDER パッケージは、フォルダ管理およびメッセージ・リスト操作機能を提供します。

次のフォルダ管理操作が可能です。

- フォルダのリスト表示
- フォルダの作成
- フォルダの削除
- フォルダ名の変更
- フォルダの完全削除

次のメッセージ・リスト操作が可能です。

- メッセージ・オブジェクトの取得
- メッセージのコピー
- メッセージ・フラグの変更

- メッセージの削除
- メッセージの検索
- メッセージのソート

MAIL_MESSAGE パッケージ

MAIL_MESSAGE パッケージは、メッセージへのアクセスおよび新規メッセージの作成に関する機能を提供します。ユーザーは、このパッケージを使用して次のことができます。

- メッセージの属性、ヘッダー、構造、内容の取得
- メッセージ・フラグの変更
- 送信または追加する新規メッセージの作成
- メッセージのハイライト、テーマ、マークアップおよびフィルタリングされたテキストの取得

MAIL_RECOVERY パッケージ

mail_recovery パッケージは、Oracle の LogMiner 機能を使用して、削除された電子メールをリカバリする機能を提供します。mail_recovery パッケージのプロシージャは次のとおりです。

- SETUP_LOGMINER
- RECOVER_MESSAGES

概念

この項では、電子メールに関連する次の概念について説明します。

- フォルダ UIDL
- メッセージ UID
- メッセージ・フラグ
- 新規および最新メッセージ
- MIME レベル

フォルダ UIDL

フォルダを作成すると、フォルダには一意の値が割り当てられ、その値が変更されないかぎり、フォルダ内のメッセージに割り当てられたメッセージの一意識別子（UID）が有効であることが保証されます。この値を、フォルダの一意識別子（UIDL）と呼びます。

メッセージ UID

フォルダ内の各メッセージには、一意識別子（UID）が割り当てられます。UID 値はセッション間で変化しないため、アプリケーションでこれらのメッセージ UID 値をキャッシュして、後でメッセージを参照することができます。メッセージ UID は、フォルダ内で常に昇順で割り当てられます。メッセージがフォルダに追加されるたびに、より大きい UID 値が割り当てられます。UID の最大許容値は、2^{32}-1、つまり 40 億です。最大値に達すると、フォルダ内のメッセージ UID は新たに割り当てられます。メッセージの削除によってメッセージ UID の連番が途切れた場合は繰り上がります。これを反映するために、フォルダの一意識別子（フォルダ UIDL）が変更されます。フォルダの一意識別子（UIDL）が変更された場合、アプリケーションはこのフォルダのメッセージ UID キャッシュを破棄して、メッセージ UID 値を新たにフェッチする必要があります。

メッセージ・フラグ

メッセージの内容は変更できませんが、メール・ユーザーはフラグ・プロパティを変更できます。メッセージがユーザーの INBOX フォルダに届いたとき、このメッセージにフラグは設定されていません。ユーザーがメッセージを開くと、メッセージには seen フラグが設定されます。

サポートされているフラグは次のとおりです。これらのフラグの使用方法はアプリケーションによって異なります。

フラグ	説明
MAIL_MESSAGE.GC_SEEN_FLAG	メッセージは既読
MAIL_MESSAGE.GC_FLAGGED_FLAG	緊急または特別な注意が必要なメッセージ
MAIL_MESSAGE.GC_ANSWERED_FLAG	メッセージに返信済
MAIL_MESSAGE.GC_DELETED_FLAG	メッセージを後で完全削除コマンドにより削除
MAIL_MESSAGE.GC_DRAFT_FLAG	メッセージは草稿状態

新規および最新メッセージ

新規メッセージと最新メッセージは、複数のクライアント・セッションが同じメール・フォルダに同時にアクセスする場合にのみ異なります。他のどのメール・クライアント・セッションもメッセージを取得していない場合、メッセージは最新とみなされます。現在のメール・ユーザー・セッションでメッセージが開かれていない場合、メッセージは新規とみなされます。

MIME レベル

メッセージの MIME レベルは、メッセージの特定のパートを識別するための文字列です。アプリケーションには、この情報は必要ありません。MIME レベルは、受け渡されるメッセージ・オブジェクトおよびボディパート・オブジェクトの一部として組み込まれます。MIME レベルは、メッセージの特定のパートを識別するために内部で使用されます。

メール・オブジェクト

Oracle Email サーバーには、3 つのメール・オブジェクトがあります。

- メール・アカウントまたはセッション
- メール・フォルダ
- メール・メッセージ

メール・アカウントには、多数のメール・フォルダが含まれます。メール・フォルダには、多数のメール・メッセージが含まれます。INBOX メール・フォルダは、メール・ユーザー・アカウントの作成時に作成される特別なフォルダです。INBOX フォルダは削除できません。すべての受信メッセージは、INBOX フォルダに届きます。

メール・ユーザーは、メール操作を実行する前に認証を受ける必要があります。MAIL_SESSION パッケージがそのための機能を提供します。ユーザーが認証されると、認証ルーチンから有効なセッション識別子が返されます。このセッション識別子は、有効な認証済セッションを必要とする他のメール操作に渡されます。メール・ユーザー・セッション識別子は、認証されたデータベース・セッションでのみ有効です。同じデータベース・セッションで複数のメール・ユーザーを認証できるように、複数のメール・ユーザー・セッションがサポートされています。認証された各セッションには、一意の識別子が関連付けられます。

フォルダ内のメッセージにアクセスするためには、ユーザーはそのフォルダを開く必要があります。1 つのメール・セッションで一度に開くことができるフォルダは 1 つのみです。フォルダを開くと、すべてのメッセージ・アクセスは現在開かれているフォルダ上で実行されます。フォルダを閉じて、別のフォルダを開くと、現在開かれているフォルダが変わります。

注意： PL/SQL API には、dbms_sql.varchar2_table 型と dbms_sql.number_table 型への参照が含まれます。

関連資料： これら 2 つの型の詳細は、オラクル社が提供するパッケージのリファレンス・マニュアルを参照してください。

MAIL_FOLDER_OBJ

MAIL_FOLDER_OBJ オブジェクトは、メール・フォルダを一意に識別します。次のように定義されます。

```
MAIL_FOLDER_OBJ(
 name  VARCHAR2(1024),
 id  NUMBER
);
MAIL_FOLDER_LIST AS TABLE OF MAIL_FOLDER_OBJ;
```

- name は、フォルダ名のフルパスです。
- id は、一意のフォルダ識別子です。
- MAIL_FOLDER_LIST は、MAIL_FOLDER_OBJ オブジェクトのネストした表です。

MAIL_FOLDER_DETAIL

MAIL_FOLDER_DETAIL オブジェクトには、特定のフォルダに関する情報が収められます。次のように定義されます。

```
MAIL_FOLDER_DETAIL (
 uidl   NUMBER,
 num_voice_recent NUMBER,
 num_fax_recent  NUMBER,
 total_recent  NUMBER,
 num_voice_unseen NUMBER,
 num_fax_unseen  NUMBER,
 total_unseen  NUMBER,
 total_msgs    NUMBER
);
```

MAIL_FOLDER_DETAIL オブジェクトには、フォルダ UIDL、メッセージ数、未読および最新のメッセージ数、未読および最新のボイスメール・メッセージと FAX メッセージに関する情報が収められます。

MAIL_SORT_CRITERIA_ELEMENT

MAIL_SORT_CRITERIA_ELEMENT オブジェクトは、ソート条件を表します。次のように定義されます。

```
MAIL_SORT_CRITERIA_ELEMENT(
 sort_header VARCHAR2(240),
 sort_order   INTEGER
);
MAIL_SORT_CRITERIA AS TABLE OF MAIL_SORT_CRITERIA_ELEMENT;
```

- sort_header は、ソートの実行に使用するメッセージ・ヘッダーです。サポートされているヘッダー名は次のとおりです。
  - CC
  - DATE
  - SUBJECT
  - SIZE
  - INTERNAL_DATE
- sort_order は、ヘッダー・フィールドを昇順または降順のいずれでソートするかを示します。値は次のとおりです。
  - MAIL_FOLDER.SORT_ASC
  - MAIL_FOLDER.SORT_DESC
- MAIL_SORT_CRITERIA は、MAIL_SORT_CRITERIA_ELEMENT オブジェクトのネストした表です。

MAIL_MESSAGE_OBJ

MAIL_MESSAGE_OBJ オブジェクトは、メール・メッセージを一意に識別します。次のように定義されます。

```
MAIL_MESSAGE_OBJ (
 folder_id NUMBER,
 msg_uid  NUMBER,
 mime_level VARCHAR2(240)
);
MAIL_MESSAGE_LIST IS TABLE OF MAIL_MESSGAGE_OBJ;
```

- folder_id は、一意のフォルダ識別子です。
- msg_uid は、フォルダ内のメッセージを一意に識別します。

- mime_level の値は、このメッセージ・オブジェクトが最上位レベルのメッセージであることを示す 0 か、含まれているメッセージ・オブジェクトであることを示す他の値です。
- MAIL_MESSAGE_LIST は、MAIL_MESSAGE_OBJ オブジェクトのネストした表です。

MAIL_BODYPART_OBJ

MAIL_BODYPART_OBJ オブジェクトは、メール・メッセージのパートを一意に識別します。次のように定義されます。

```
MAIL_BODYPART_OBJ (
 content_type VARCHAR2(240),
 mime_level VARCHAR2(240),
 folder_id NUMBER,
 msg_uid  NUMBER,
 smime_ind NUMBER
);
MAIL_BODYPART_LIST IS TABLE OF MAIL_BODYPART_OBJ;
```

- folder_id は、一意のフォルダ識別子です。
- msg_uid は、フォルダ内のメッセージを一意に識別します。
- mime_level は、メッセージの特定のボディパートを識別します。
- content_type には、このボディパートのメインの Content-Type ヘッダー値が含まれます。
- smime_ind の値は 1 で、ボディパートが復号化されたメッセージからのものであることを示します。
- mail_bodypart_list は、MAIL_BODYPART_OBJ オブジェクトのネストした表です。

VARCHAR2_TABLE

VARCHAR2_TABLE は、次のように、最大長が 2000 文字の可変長文字列のネストした表を定義します。

```
VARCHAR2_TABLE IS TABLE OF VARCHAR2(2000);
```

MAIL_HEADER_OBJ

MAIL_HEADER_OBJ オブジェクトは、メッセージ・ヘッダー名と値のペアの保存に使用されます。次のように定義されます。

```
MAIL_HEADER_OBJ (
 header_prompt      VARCHAR2(1000),
 header_value       VARCHAR2_TABLE
);
```

- header_prompt は、ヘッダーの名前です。
- header_value は、ヘッダーの値です。これは、2000 文字の可変長文字列の表 VARCHAR2_TABLE です。ヘッダー値の長さが 2000 文字を超すと、2000 文字長の複数の文字列に分割されます。
- MAIL_HEADER_LIST は、MAIL_HEADER_OBJ オブジェクトのネストした表です。

MAIL_RECOVERY パッケージ

mail_recovery パッケージは、Oracle の LogMiner 機能を使用して、削除された電子メールをリカバリするために提供されています。

REDO リスト・ファイルは、mail_recovery パッケージで REDO ログのリストを取得するために使用されます。このファイルは管理者が用意する必要があり、テキスト・エディタを使用して手動で、または UNIX や Windows NT のディレクトリ・リスト・コマンドを出力して作成できます。

ファイルには、REDO ログ・ファイル名がフルパスで含まれ、1 行に 1 つずつリストされている必要があります。REDO ログには、オンラインの REDO ログ、アーカイブされたログ、またはその両方を使用できます。

init.ora パラメータ UTL_FILE_DIR が、REDO リスト・ファイルにアクセスするように設定されていることを確認してください。次に例を示します。

```
/oracle/database/redo01.log
/oracle/database/redo02.log
/oracle/database/redo03.log
```

MAIL_RECOVERY パッケージには、次のプロシージャが含まれます。

- SETUP_LOGMNR プロシージャ
- RECOVER_MESSAGES プロシージャ

SETUP_LOGMNR プロシージャ

setup_logmnr プロシージャは、LogMiner をリカバリのために初期化します。

構文

```
PROCEDURE setup_logmnr(
 p_dictionary_filename IN VARCHAR2,
 p_redolist_location   IN VARCHAR2,
 p_redolist_filename   IN VARCHAR2
 p_starttime           IN DATE DEFAULT '01-jan-1988',
 p_endtime             IN DATE DEFAULT '01-jan-2099');
```

パラメータ

パラメータ	説明
p_dictionary_filename	LogMiner のデータ・ディクショナリ・ファイルのフルパス付きの名前。
p_redolist_location	REDO リスト・ファイルのディレクトリの場所。
p_redolist_filename	REDO リスト・ファイルのファイル名。
p_starttime	REDO リストの開始時間。指定された開始時間と同じかそれより遅いタイムスタンプの REDO レコードのみを考慮します。
p_endtime	REDO リストの終了時間。指定された終了時間と同じかそれより早いタイムスタンプの REDO レコードのみを考慮します。

RECOVER_MESSAGES プロシージャ

recover_messages プロシージャは、ユーザーのかわりにリカバリを実行し、指定されたフォルダにメッセージをリストアします。

構文

```
PROCEDURE recover_messages(
  p_domainname IN  VARCHAR2,
  p_username   IN  VARCHAR2,
  p_foldername IN  VARCHAR2,
  p_autocommit IN BOOLEAN DEFAULT TRUE);
```

パラメータ	説明
p_domainname	ユーザーのドメイン名。
p_username	リカバリを実行する Oracle Email ユーザー名。
p_foldername	リカバリしたメッセージをリストアするフォルダ名。NULL の場合、NULL が渡され、recover_messages は RECMSG_current_date_time という名前の新しいフォルダを作成します。
p_autocommit	True の場合、recover_messages 内で頻繁にコミットが実行されます。 False の場合、recover_messages プロシージャ内でコミットは実行されません。end_logmnr プロシージャによって、LogMiner セッションが終了します。

MAIL_SESSION パッケージ

MAIL_SESSION パッケージは、ユーザー認証とログアウト機能を提供します。ユーザーは、MAIL_SESSION.login() を複数回コールして、同じデータベース・セッションを使用する複数のメール・セッションを作成できます。各セッション ID は、1 つの有効なメール・セッションを示します。

MAIL_SESSION パッケージには、次のプロシージャとファンクションが含まれます。

- LOGIN プロシージャ
- LOGOUT プロシージャ
- GET_CURRENT_USAGE プロシージャ
- IS_OVER_QUOTA ファンクション

LOGIN プロシージャ

このプロシージャは、ユーザーのユーザー名とパスワードを使用して、ユーザーを認証します。

発生する例外

mail_errors.login_err

構文

```
PROCEDURE login (
user_name   IN VARCHAR2,
password   IN  VARCHAR2,
domain   IN  VARCHAR2,
ldap_host   IN  VARCHAR2,
session_id   OUT  NUMBER,
ldap_port  IN  NUMBER DEFAULT 389
);
PROCEDURE login (
user_address  IN  VARCHAR2,
password   IN  VARCHAR2,
ldap_host   IN  VARCHAR2,
session_id   OUT  NUMBER,
ldap_port   IN  NUMBER DEFAULT 389
);
```

パラメータ

パラメータ	説明
user_name	ドメイン部分のないユーザー・アカウント名
password	ユーザー・パスワード
user_address	ユーザーのインターネット・アドレス： *user_name@domain*
domain	ユーザー・ドメイン
ldap_host	Oracle Internet Directory が構成されているホスト名
ldap_port	Oracle Internet Directory がリスニングするポートで、デフォルトは 389
session_id	このユーザーの認証済セッションを表す一意の識別子

LOGOUT プロシージャ

このプロシージャは、このユーザー・セッションに関連付けられているすべてのリソースを解放します。

構文

```
PROCEDURE logout (
session_id   IN  NUMBER
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子

GET_CURRENT_USAGE プロシージャ

このプロシージャは、ユーザーの現在の領域使用量を返します。

発生する例外

mail_errors.unauthenticated_err

構文

```
PROCEDURE get_current_usage (
session_id         IN  NUMBER,
usage      OUT NUMBER
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
usage	ユーザーの現在の領域使用量（バイト単位）

IS_OVER_QUOTA ファンクション

このファンクションは、ユーザーが現在クォータを超えている場合は True を返し、超えていない場合は False を返します。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
FUNCTION is_over_quota (
session_id    IN  NUMBER
) RETURN boolean;
FUNCTION is_over_quota (
session_id    IN  NUMBER,
quota         OUT NUMBER,
usage         OUT NUMBER
) RETURN boolean;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
quota	ユーザーのクォータ（バイト単位）
usage	ユーザーの現在の領域使用量（バイト単位）

MAIL_FOLDER パッケージ

MAIL_FOLDER パッケージは、フォルダ関連の機能を提供します。操作の前にセッションの妥当性がチェックされます。検索およびソート機能は、IMAP4 プロトコルに基づきます。検索機能には、Oracle Text ベースの検索が含まれます。

MAIL_FOLDER には、次のプロシージャとファンクションが含まれます。

- GET_FOLDER_OBJ プロシージャ
- LIST_TOPLEVEL_FOLDERS プロシージャ
- LIST_FOLDERS プロシージャ
- LIST_TOPLEVEL_SUBDFLDRS プロシージャ
- LIST_SUBSCRIBED_FOLDERS プロシージャ
- IS_FOLDER_SUBSCRIBED ファンクション

GET_FOLDER_OBJ プロシージャ

このプロシージャは、指定されたフォルダ名のフォルダ・オブジェクトを返します。フォルダがメール・ストアに存在しない場合は、FOLDER_NOT_FOUND 例外が発生します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_not_found_err
```

構文

```
PROCEDURE get_folder_obj (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2,
folder_obj   OUT  MAIL_FOLDER_OBJ
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス
folder_obj	返される MAIL_FOLDER_OBJ

LIST_TOPLEVEL_FOLDERS プロシージャ

このプロシージャは、最上位フォルダ・オブジェクトのリストを返します。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE list_toplevel_folders (
session_id   IN  NUMBER,
folder_list   OUT  MAIL_FOLDER_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_list	最上位フォルダ・オブジェクトのリスト

LIST_FOLDERS プロシージャ

このプロシージャは、指定された親フォルダの直下のフォルダ・オブジェクトのリストを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_not_found_err
```

構文

```
PROCEDURE list_folders (
session_id   IN  NUMBER,
parent_name  IN  VARCHAR2,
folder_list   OUT  MAIL_FOLDER_LIST
);
PROCEDURE list_folders (
session_id   IN  NUMBER,
parent_obj   IN  MAIL_FOLDER_OBJ,
folder_list   OUT  MAIL_FOLDER_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
parent_name	親フォルダのフルパス
parent_obj	親フォルダを表す MAIL_FOLDER_OBJ
folder_list	親フォルダの下にある子フォルダ・オブジェクトのリスト

LIST_TOPLEVEL_SUBDFLDRS プロシージャ

このプロシージャは、最上位レベルの登録済フォルダのリストを返します。

発生する例外

mail_errors.unauthenticated_err

構文

```
PROCEDURE list_toplevel_subdfldrs (
session_id   IN  NUMBER,
foldername_list  OUT  DBMS_SQL.VARCHAR2_TABLE
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
foldername_list	最上位レベルの登録済フォルダのリスト

LIST_SUBSCRIBED_FOLDERS プロシージャ

このプロシージャは、指定された親フォルダの登録済の子フォルダのリストを返します。

発生する例外

mail_errors.unauthenticated_err

構文

```
PROCEDURE list_subscribed_folders (
session_id   IN  NUMBER,
parent_name  IN  VARCHAR2,
foldername_list  OUT  DBMS_SQL.VARCHAR2_TABLE
);
PROCEDURE list_subscribed_folders (
session_id   IN  NUMBER,
parent_obj   IN  MAIL_FOLDER_OBJ,
foldername_list  OUT  DBMS_SQL.VARCHAR2_TABLE
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
parent_name	親フォルダ名のフルパス
parent_obj	親フォルダを表す MAIL_FOLDER_OBJ
foldername_list	親フォルダの下にある登録済の子フォルダ・オブジェクトのリスト

IS_FOLDER_SUBSCRIBED ファンクション

このファンクションは、フォルダが登録されているかどうかをテストします。

発生する例外

mail_errors.unauthenticated_err

構文

```
FUNCTION is_folder_subscribed (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2
) return BOOLEAN;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス

SUBSCRIBE_FOLDER プロシージャ

このプロシージャは、指定されたフォルダを登録します。フォルダがすでに登録されている場合でもエラーは返されません。

発生する例外

mail_errors.unauthenticated_err

構文

```
PROCEDURE subscribe_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス

UNSUBSCRIBE_FOLDER プロシージャ

このプロシージャは、指定されたフォルダの登録を解除します。コールの時点でフォルダが登録されていなくてもエラーは返されません。

発生する例外

mail_errors.unauthenticated_err

構文

```
PROCEDURE unsubscribe_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス

HAS_FOLDER_CHILDREN ファンクション

このファンクションは、子フォルダが存在するかどうかをテストします。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
FUNCTION has_folder_children (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ
) return BOOLEAN;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_obj	フォルダ・オブジェクト

GET_FOLDER_DETAILS プロシージャ

このプロシージャは、フォルダ UIDL 識別子、総メッセージ数、未読メッセージ数、最新メッセージ数などのフォルダ情報を返します。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE get_folder_details (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
folder_detail_obj OUT MAIL_FOLDER_DETAIL
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_obj	フォルダ・オブジェクト
folder_detail_obj	フォルダに関する情報を含むフォルダ詳細オブジェクト

CREATE_FOLDER プロシージャ

このプロシージャは、指定された名前のフォルダを作成し、そのフォルダを表すフォルダ・オブジェクトを返します。親フォルダでサブフォルダを作成できない場合、FOLDER_TYPE_ERR が返されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_already_exists_err
mail_errors.folder_type_err
```

構文

```
PROCEDURE create_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2,
folder_obj   OUT  MAIL_FOLDER_OBJ
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス
folder_obj	返される MAIL_FOLDER_OBJ

DELETE_FOLDER プロシージャ

このプロシージャは、指定されたフォルダを削除します。再帰フラグが True に設定されている場合、フォルダ内のすべてのメッセージ、すべてのサブフォルダとそのメッセージ、およびフォルダ自体が削除されます。再帰フラグが False に設定されている場合、次のアクションが実行されます。

- サブフォルダが存在しない場合、フォルダ内のメッセージとフォルダ自体が削除されます。
- サブフォルダが存在する場合、フォルダ内のメッセージが削除され、フォルダは選択不可としてマークされます（このフォルダで OPEN_FOLDER 操作を実行すると失敗します)。
- 選択不可としてマークされたフォルダで DELETE_FOLDER プロシージャをコールし、そのフォルダにサブフォルダが含まれている場合、MAIL_ERRORS.FOLDER_TYPE_ERR が発生します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.operation_not_allowed
mail_errors.folder_type_err
mail_errors.folder_not_found_err
```

構文

```
PROCEDURE delete_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2,
recursive  IN BOOLEAN DEFAULT false
);
PROCEDURE delete_folder (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
recursive  IN BOOLEAN DEFAULT false
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス
folder_obj	フォルダを表す MAIL_FOLDER_OBJ
recursive	True の場合、フォルダとサブフォルダを削除

RENAME_FOLDER プロシージャ

このプロシージャは、指定されたフォルダの名前を変更し、新しいフォルダ・オブジェクトを返します。INBOX の名前を変更すると、INBOX にあるすべてのメッセージが新しいフォルダに移動し、INBOX フォルダは空になります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_not_found_err
mail_errors.folder_already_exists_err
mail_errors.operation_not_allowed
```

構文

```
PROCEDURE rename_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2,
new_folder_name  IN  VARCHAR2,
folder_obj   OUT  MAIL_FOLDER_OBJ
);
PROCEDURE rename_folder (
session_id   IN  NUMBER,
folder_obj   IN OUT MAIL_FOLDER_OBJ,
new_folder_name  IN  VARCHAR2
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス
new_folder_name	フォルダの新しい名前
folder_obj	フォルダを表す MAIL_FOLDER_OBJ

OPEN_FOLDER プロシージャ

このプロシージャは、指定されたフォルダを開き、フォルダが正常に開いた場合、フォルダ・オブジェクトを返します。フォルダに NOSELECT フラグが設定されている場合、MAIL_ERRORS.FOLDER_TYPE_ERR が発生します。このコールの前に別のフォルダがすでに開かれている場合、EXPUNGE を実行せずに閉じます（DELETED メッセージ・フラグ付きのメッセージは、このフォルダから削除されません）。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_not_found_err
mail_errors.folder_type_err
```

構文

```
PROCEDURE open_folder (
session_id   IN  NUMBER,
folder_name   IN  VARCHAR2,
folder_obj   OUT  MAIL_FOLDER_OBJ
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_name	フォルダ名のフルパス
folder_obj	フォルダを表す MAIL_FOLDER_OBJ

GET_FOLDER_MESSAGES プロシージャ

このプロシージャは、現在開かれているフォルダにある指定されたメッセージ・タイプのすべてのメッセージを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE get_folder_messages (
session_id   IN  NUMBER,
message_list  OUT MAIL_MESSAGE_LIST,
message_type IN NUMBER DEFAULT MAIL_MESSAGE.GC_ALL_MAIL
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_list	フォルダに属するメッセージ・オブジェクトのリスト。
message_type	取得するメッセージのタイプ。デフォルトではすべてのタイプを取得します。メッセージ・タイプは、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_ALL_MAIL ■ MAIL_MESSAGE.GC_EMAIL ■ MAIL_MESSAGE.GC_VOICE_MAIL ■ MAIL_MESSAGE.GC_FAX_MAIL ■ MAIL_MESSAGE.GC_NEWS_MAIL

GET_MESSAGE プロシージャ

このプロシージャは、現在開かれているフォルダで指定されたメッセージ UID に対応するメッセージ・オブジェクトを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE get_message (
session_id   IN  NUMBER,
message_uid  IN NUMBER,
message_obj  OUT MAIL_MESSAGE_OBJ
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_uid	メッセージ識別子
message_obj	指定された UID に対応する MAIL_MESSAGE_OBJ タイプ

CLOSE_FOLDER プロシージャ

このプロシージャは、現在開かれているフォルダを閉じます。EXPUNGE_FLAG が True に設定されている場合、フォルダ内にある Deleted フラグが設定されたすべてのメッセージが削除されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE close_folder (
session_id   IN  NUMBER,
expunge_flag IN BOOLEAN
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
expunge_flag	閉じる前にフォルダを完全削除するかどうかを示すフラグ

GET_MSG_FLAGS プロシージャ

このプロシージャは、現在開かれているフォルダで指定されたメッセージ・リストに属するメッセージ・フラグを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE get_msg_flags (
session_id   IN  NUMBER,
message_list  IN MAIL_MESSAGE_LIST,
message_flags OUT DBMS_SQL.NUMBER_TABLE
);
PROCEDURE get_msg_flags (
session_id   IN  NUMBER,
message_uid_list IN DBMS_SQL.NUMBER_TABLE,
message_flags OUT DBMS_SQL.NUMBER_TABLE
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_list	フォルダに属するメッセージ・オブジェクトのリスト。
message_uid_list	フォルダ内のメッセージを識別するメッセージ UID のリスト。

パラメータ	説明
message_flags	リクエストされたメッセージのリストに対応するメッセージ・フラグのリスト。フラグの値は、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEEN_FLAG ■ MAIL_MESSAGE.GC_FLAGGED_FLAG ■ MAIL_MESSAGE.GC_ANSWERED_FLAG ■ MAIL_MESSAGE.GC_DELETED_FLAG ■ MAIL_MESSAGE.GC_DRAFT_FLAG

SET_MSG_FLAGS プロシージャ

このプロシージャは、現在開かれているフォルダで指定されたメッセージ・リストに属するメッセージ・フラグを設定または設定解除します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE set_msg_flags (
session_id   IN  NUMBER,
message_list  IN MAIL_MESSAGE_LIST,
flags  IN NUMBER,
set_flag  IN BOOLEAN
);
PROCEDURE set_msg_flags (
session_id   IN  NUMBER,
message_uid_list IN DBMS_SQL.NUMBER_TABLE,
flags  IN NUMBER,
set_flag  IN BOOLEAN
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_list	フォルダに属するメッセージ・オブジェクトのリスト。
message_uid_list	フォルダ内のメッセージを識別するメッセージ UID のリスト。
flags	リクエストされたメッセージのリストに対応するメッセージ・フラグのリスト。フラグの値は、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEEN_FLAG ■ MAIL_MESSAGE.GC_FLAGGED_FLAG ■ MAIL_MESSAGE.GC_ANSWERED_FLAG ■ MAIL_MESSAGE.GC_DELETED_FLAG ■ MAIL_MESSAGE.GC_DRAFT_FLAG
set_flag	True の場合、flags の値を設定します。False の場合、flags の値の設定を解除します。

DELETE_MESSAGES プロシージャ

このプロシージャは、現在開かれているフォルダで指定されたメッセージのリストを削除します。これは、メッセージに削除のマークを付け、フォルダで完全削除の操作を実行することと同じです。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE delete_messages (
session_id   IN  NUMBER,
message_uid_list IN DBMS_SQL.NUMBER_TABLE
);
PROCEDURE delete_messages (
session_id        IN  NUMBER,
message_list      IN  MAIL_MESSAGE_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_uid_list	フォルダ内のメッセージを識別するメッセージ UID のリスト
message_list	フォルダに属するメッセージ・オブジェクトのリスト

EXPUNGE_FOLDER プロシージャ

このプロシージャは、現在開かれているフォルダ内の gc_deleted_flag フラグが設定されているすべてのメッセージを削除します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE expunge_folder (
session_id   IN  NUMBER,
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子

IS_FOLDER_OPEN ファンクション

このファンクションは、フォルダがユーザーのセッションで現在選択されているフォルダと同じかどうかをテストします。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
FUNCTION is_folder_open (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ
) return BOOLEAN;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_obj	フォルダ・オブジェクト

CHECK_NEW_MESSAGES ファンクション

このファンクションは、現在選択されているフォルダに新規メッセージがあるかどうかをテストします。新規メッセージとは、最後に GET_NEW_MESSAGES をコールして以降フォルダで開かれていないメッセージです。フォルダを最初に開いたときに、取得済とみなされる最後のメッセージは、任意のメール・クライアントで最後に開かれたメッセージです。その後、最後のメッセージは、GET_FOLDER_MESSAGES と GET_NEW_MESSAGES のコールによって変わります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
FUNCTION check_new_messages (
session_id   IN  NUMBER,
message_type IN  NUMBER DEFAULT MAIL_MESSAGE.GC_ALL_MAIL
) return BOOLEAN;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_type	取得するメッセージのタイプ。デフォルトではすべてのタイプを取得します。メッセージ・タイプは、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_ALL_MAIL ■ MAIL_MESSAGE.GC_EMAIL ■ MAIL_MESSAGE.GC_VOICE_MAIL ■ MAIL_MESSAGE.GC_FAX_MAIL ■ MAIL_MESSAGE.GC_NEWS_MAIL

CHECK_RECENT_MESSAGES ファンクション

このファンクションは、指定したフォルダに最新メッセージがあるかどうかをテストします。最新メッセージとは、どのメール・クライアントもまだ取得していないメッセージです。このプロシージャは、閉じたフォルダでコールできます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_not_found_err
```

構文

```
FUNCTION check_recent_messages (
session_id   IN  NUMBER,
folder_name  IN VARCHAR2,
message_type IN  NUMBER DEFAULT MAIL_MESSAGE.GC_ALL_MAIL
) return BOOLEAN;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
folder_name	フォルダ名のフルパス。

パラメータ	説明
message_type	取得するメッセージのタイプ。デフォルトではすべてのタイプを取得します。メッセージ・タイプは、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_ALL_MAIL ■ MAIL_MESSAGE.GC_EMAIL ■ MAIL_MESSAGE.GC_VOICE_MAIL ■ MAIL_MESSAGE.GC_FAX_MAIL ■ MAIL_MESSAGE.GC_NEWS_MAIL

GET_NEW_MESSAGES プロシージャ

このプロシージャは、現在選択されているフォルダにあるすべての新規メッセージを返します。新規メッセージとは、最後にメッセージを取得して以降フォルダで開かれていないメッセージです。フォルダを最初に開いたときに、取得済とみなされる最後のメッセージは、任意のメール・クライアントで最後に開かれたメッセージです。その後、最後のメッセージは、GET_FOLDER_MESSAGES と GET_NEW_MESSAGES のコールによって変わります。MESSAGE_TYPE を指定すると、指定したタイプのメッセージだけが返されます。

発生する例外

mail_errors.unauthenticated_err
mail_errors.folder_closed_err

構文

```
PROCEDURE get_new_messages (
session_id   IN  NUMBER,
message_list  OUT MAIL_MESSAGE_LIST,
message_type IN  NUMBER DEFAULT MAIL_MESSAGE.GC_ALL_MAIL
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_list	新規メッセージ・オブジェクトのリスト。

パラメータ	説明
message_type	取得するメッセージのタイプ。デフォルトではすべてのタイプを取得します。メッセージ・タイプは、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_ALL_MAIL ■ MAIL_MESSAGE.GC_EMAIL ■ MAIL_MESSAGE.GC_VOICE_MAIL ■ MAIL_MESSAGE.GC_FAX_MAIL ■ MAIL_MESSAGE.GC_NEWS_MAIL

COPY_MESSAGES プロシージャ

このプロシージャは、現在選択されているフォルダ内のメッセージを別のフォルダにコピーします。宛先フォルダに NOSELECT フラグが設定されている場合、MAIL_ERRORS.FOLDER_TYPE_ERR 例外が発生します。指定したメッセージが現在開かれているフォルダに属していない場合、MAIL_ERRORS.PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
mail_errors.folder_not_found_err
mail_errors.folder_type_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE copy_messages (
session_id   IN  NUMBER,
message_list  IN MAIL_MESSAGE_LIST,
to_folder_name IN  VARCHAR2
);
PROCEDURE copy_messages (
session_id   IN  NUMBER,
message_uid_list IN DBMS_SQL.NUMBER_TABLE,
to_folder_name IN  VARCHAR2
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_list	メッセージ・オブジェクトのリスト
message_uid_list	メッセージ UID のリスト
to_folder_name	宛先フォルダ名のフルパス

IS_FOLDER_MODIFIED ファンクション

このファンクションは、現在選択されているフォルダ内のメッセージが別のセッションから変更されているかどうかをテストします。フォルダで別のクライアントまたはセッションによってメッセージ・フラグが変更されていたり、メッセージが削除されている場合、そのフォルダは変更されています。フォルダが変更された場合、ユーザーは、GET_NEW_MESSAGES プロシージャを再発行して、メール・ストアと同期させる必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
FUNCTION is_folder_modified (
session_id   IN  NUMBER
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子

SORT_FOLDER プロシージャ

このプロシージャは、指定されたソート条件でフォルダをソートし、メッセージ UID のソート済のリストを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE sort_folder (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
sort_criteria  IN MAIL_SORT_CRITERIA,
message_uid_list OUT DBMS_SQL.NUMBER_TABLE
);
PROCEDURE sort_folder (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
sort_criteria  IN MAIL_SORT_CRITERIA,
message_list  OUT MAIL_MESSAGE_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
folder_obj	フォルダを表す MAIL_FOLDER_OBJ。
sort_criteria	ソート条件のリスト。 値は次のとおりです。 ■ Subject ■ Cc ■ From ■ Date ■ Internal_date ■ Size
message_list	メッセージ・オブジェクトの順序付けされたリスト。
message_uid_list	メッセージ UID の順序付けされたリスト。

SEARCH_FOLDER プロシージャ

このプロシージャは、フォルダを検索し、指定された条件を満たすメッセージ・オブジェクトのリストを返します。検索条件の形式は、メッセージ・セット内の検索を指定した場合、セットがパラメータとして渡される点を除いて、IMAP4 プロトコル [RFC 2060] の形式と同じです。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE search_folder (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
search_criteria IN VARCHAR2,
message_list  OUT MAIL_MESSAGE_LIST
);
PROCEDURE search_folder (
session_id   IN  NUMBER,
folder_obj   IN  MAIL_FOLDER_OBJ,
search_criteria IN VARCHAR2,
in_message_list IN MAIL_MESSAGE_LIST,
message_list  OUT MAIL_MESSAGE_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
folder_obj	フォルダを表す MAIL_FOLDER_OBJ
search_criteria	IMAP4 標準に従った検索条件のリスト
in_message_list	検索するメッセージ・オブジェクトのリスト
message_list	検索条件を満たすメッセージ・オブジェクトのリスト

MAIL_MESSAGE パッケージ

MAIL_MESSAGE パッケージは、メッセージの取得、メッセージの作成および Oracle Text 関連の機能を提供します。送信メッセージを作成する場合を除き、操作の実行前にセッションの妥当性がチェックされます。ユーザーは、一度に 1 つのメッセージのみを作成でき、違反が発生すると MSG_COMPOSE_LIMIT_ERR が発生します。

MAIL_MESSAGE パッケージには、次のメッセージ取得プロシージャが含まれます。

- GET_MESSAGE_OBJ プロシージャ
- GET_INCLUDED_MESSAGE プロシージャ
- GET_HEADER プロシージャ
- GET_HEADERS プロシージャ
- GET_CONTENT_TYPE プロシージャ
- GET_REPLY_TO プロシージャ
- GET_SENT_DATE プロシージャ
- GET_SUBJECT プロシージャ
- GET_FROM プロシージャ
- GET_MESSAGEID プロシージャ
- GET_CONTENTID プロシージャ
- GET_CONTENTLANG プロシージャ
- GET_COTENTMD5 プロシージャ
- GET_CHARSET プロシージャ
- GET_CONTENTDISP プロシージャ
- GET_ENCODING プロシージャ
- GET_CONTENT_FILENAME プロシージャ
- GET_MSG_SIZE プロシージャ
- GET_RECEIVED_DATE プロシージャ
- GET_BODYPART_SIZE プロシージャ
- GET_CONTENT_LINECOUNT プロシージャ
- GET_MULTIPART_BODYPARTS プロシージャ
- GET_MSG プロシージャ
- GET_MSG_BODY プロシージャ

- GET_BODYPART_CONTENT プロシージャ
- GET_MSGS_FLAGS プロシージャ
- SET_MSGS_FLAGS プロシージャ
- GET_AUTH_INFO プロシージャ

MAIL_MESSAGE パッケージには、次のメッセージ作成プロシージャが含まれます。

- COMPOSE_MESSAGE プロシージャ
- SET_MSGHEADER プロシージャ
- SET_BPHEADER プロシージャ
- SET_HEADER プロシージャ
- ADD_BODYPART プロシージャ
- ADD_INCLMSG_BODYPART プロシージャ
- SET_INCLMSG_BODYPART プロシージャ
- SET_CONTENT プロシージャ
- SEND_MESSAGE プロシージャ
- APPEND_MESSAGE プロシージャ
- DECRYPT_MESSAGE プロシージャ
- VERIFY_MESSAGE プロシージャ

MAIL_MESSAGE パッケージには、次の Oracle Text プロシージャが含まれます。

- GET_THEMES プロシージャ
- GET_HIGHLIGHT プロシージャ
- GET_MARKUPTEXT プロシージャ
- GET_FILTERED_TEXT プロシージャ
- GET_TOKENS プロシージャ

GET_MESSAGE_OBJ プロシージャ

このプロシージャは、現在開かれているフォルダにある、指定されたメッセージ UID のメッセージ・オブジェクトを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE get_message_obj (
session_id   IN  NUMBER,
message_uid  IN  NUMBER,
message_obj  OUT MAIL_MESSAGE_OBJ);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_uid	メッセージ UID
message_obj	返される MAIL_MESSAGE_OBJ

GET_INCLUDED_MESSAGE プロシージャ

このプロシージャは、含まれているメッセージを表すメッセージ・オブジェクトを返します。指定するメッセージ・オブジェクトまたはボディパート・オブジェクトのメッセージ・タイプは Content-Type である必要があります。それ以外の場合、MAIL_ERRORS.PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.param_parse_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_included_message (
session_id       IN  NUMBER,
message_obj      IN  MAIL_MESSAGE_OBJ,
incl_message_obj OUT MAIL_MESSAGE_OBJ);
```

```
PROCEDURE get_included_message (
session_id       IN  NUMBER,
bodypart_obj     IN MAIL_BODYPART_OBJ,
incl_message_obj OUT MAIL_MESSAGE_OBJ);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	Content-Type メッセージのメッセージ・オブジェクト
bodypart_obj	Content-Type メッセージのボディパート・オブジェクト
incl_message_obj	返される MAIL_MESSAGE_OBJ

GET_HEADER プロシージャ

このプロシージャは、指定されたヘッダー・プロンプトに対応するヘッダー値を返します。

メッセージ・オブジェクトが渡された場合、ヘッダー値は最上位レベルのメッセージ・ヘッダーを表します。

ボディパート・オブジェクトが渡された場合、ヘッダー値はそのボディパート・ヘッダーを表します。

ヘッダーにヘッダー・プロンプトがない場合、NULL 値が返されます。戻り値が VARCHAR2 型で、値が 2000 文字よりも長い場合、2000 文字に切り捨てられます。同じプロンプトのヘッダーが複数ある場合、戻り値は最初のヘッダーになります。戻り値が MAIL_HEADER_LIST オブジェクト・リストの場合、指定されたプロンプト名のすべてのヘッダーが返されます。名前と値の各ペアは、MAIL_HEADER_OBJ オブジェクトで表されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_header (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
header_prompt IN  VARCHAR2,
header_value  OUT VARCHAR2);
PROCEDURE get_header (
```

```
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
header_prompt IN  VARCHAR2,
header_value  OUT VARCHAR2);
PROCEDURE get_header (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
header_list   OUT MAIL_HEADER_LIST);
PROCEDURE get_header (
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
header_list   OUT MAIL_HEADER_LIST);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
header_prompt	メッセージ・ヘッダー
header_value	対応するヘッダー値
header_list	指定されたプロンプトを持つヘッダー・オブジェクトのリスト

GET_HEADERS プロシージャ

このプロシージャは、指定されたメッセージ・パートのすべてのヘッダー値を返します。メッセージ・オブジェクトが渡された場合、ヘッダー値は最上位レベルのメッセージ・ヘッダーを表します。

ボディパート・オブジェクトが渡された場合、ヘッダー値はそのボディパート・ヘッダーを表します。

戻り値が dbms_sql.varchar2_table 型で、値が 2000 文字よりも長い場合、すべてのヘッダー値は 2000 文字に切り捨てられます。ヘッダーが MAIL_HEADER_LIST オブジェクト・リストで返される場合、名前と値の各ペアは MAIL_HEADER_OBJ オブジェクトで表されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_headers (
session_id     IN  NUMBER,
message_obj    IN  MAIL_MESSAGE_OBJ,
header_prompts IN  dbms_sql.varchar2_table,
header_values  OUT dbms_sql.varchar2_table);
PROCEDURE get_headers (
session_id     IN  NUMBER,
bodypart_obj   IN  MAIL_BODYPART_OBJ,
header_prompts IN  dbms_sql.varchar2_table,
header_values  OUT dbms_sql.varchar2_table);
PROCEDURE get_headers (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
header_list  OUT MAIL_HEADER_LIST);
PROCEDURE get_headers (
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
header_list   OUT MAIL_HEADER_LIST);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
header_prompts	このパートに属するヘッダー名の配列
header_values	対応するヘッダー値の配列
header_list	このパートに属するヘッダー・オブジェクトのリスト

GET_CONTENT_TYPE プロシージャ

このプロシージャは、Content-Type ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_content_type (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
content_type OUT VARCHAR2);
PROCEDURE get_content_type (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
content_type OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
content_type	Content-Type ヘッダー値

GET_REPLY_TO プロシージャ

このプロシージャは、Reply-To ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_reply_to (
session_id  IN  NUMBER,
message_obj IN  MAIL_MESSAGE_OBJ,
replyTo_str OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
replyTo_str	Reply-To ヘッダー値

GET_SENT_DATE プロシージャ

このプロシージャは、Date ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_sent_date (
session_id  IN  NUMBER,
message_obj IN  MAIL_MESSAGE_OBJ,
sent_date   OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
sent_date	Date ヘッダー値

GET_SUBJECT プロシージャ

このプロシージャは、Subject ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_subject (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
subject_str  OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
subject_str	Subject ヘッダー値

GET_FROM プロシージャ

このプロシージャは、From ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_from (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
from_str     OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
from_str	From ヘッダー値

GET_MESSAGEID プロシージャ

このプロシージャは、Message-ID ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_messageID (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
messageID_str OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
messageID_str	Message-ID ヘッダー値

GET_CONTENTID プロシージャ

このプロシージャは、Content-ID ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_contentID (
session_id     IN  NUMBER,
bodypart_obj   IN  MAIL_BODYPART_OBJ,
contentID_str  OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
contentID_str	Content-ID ヘッダー値

GET_CONTENTLANG プロシージャ

このプロシージャは、Content-Language ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_contentLang (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
language     OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
language	Content-Language ヘッダー値

GET_COTENTMD5 プロシージャ

このプロシージャは、Content-MD5 ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_contentMD5 (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
md5          OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
md5	Content-MD5 ヘッダー値

GET_CHARSET プロシージャ

このプロシージャは、Content-Type ヘッダー値を取得し、キャラクタ・セット属性値を抽出するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_charset (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
charset      OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
charset	キャラクタ・セット属性値

GET_CONTENTDISP プロシージャ

このプロシージャは、Content-Disposition ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_contentDisp (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
disposition  OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
disposition	Content-Disposition ヘッダー値

GET_ENCODING プロシージャ

このプロシージャは、Content-Transfer-Encoding ヘッダー値を取得するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_encoding (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
encoding     OUT VARCHAR2);
PROCEDURE get_encoding (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
encoding     OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
encoding	Content-Transfer-Encoding ヘッダー値

GET_CONTENT_FILENAME プロシージャ

このプロシージャは、Content-Disposition ヘッダー値を取得し、ファイル名属性値を抽出するために使用します。特定のヘッダー・プロンプトを使用して、内部で GET_HEADER プロシージャをコールします。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_content_filename (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
filename     OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
filename	ファイル名属性値

GET_MSG_SIZE プロシージャ

このプロシージャは、メッセージのサイズを返します。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE get_msg_id (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
message_size OUT NUMBER);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
message_size	メッセージ・サイズ

GET_RECEIVED_DATE プロシージャ

このプロシージャは、メール・ストアでメッセージを受信した時間を取得します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
```

構文

```
PROCEDURE get_rcvd_date (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
date_format  IN  VARCHAR2,
date_str     OUT VARCHAR2);
PROCEDURE get_rcvd_date (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
received_date OUT DATE);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
date_format	日付の書式
date_str	指定した文字列書式の受信日
received_date	Oracle DATE 型の受信日

GET_BODYPART_SIZE プロシージャ

このプロシージャは、ボディパートのサイズを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
```

構文

```
PROCEDURE get_bodypart_size (
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
bodypart_size OUT NUMBER);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
bodypart_size	ボディパート・サイズ

GET_CONTENT_LINECOUNT プロシージャ

このプロシージャは、ボディパートの行数を返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
```

構文

```
PROCEDURE get_content_linecount (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
line_count   OUT NUMBER);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
line_count	ボディパートの総行数

GET_MULTIPART_BODYPARTS プロシージャ

このプロシージャは、指定されたマルチパート・メッセージまたはボディパートに属するボディパートのリストを返します。渡されたメッセージ・オブジェクトまたはボディパート・オブジェクトがマルチパート MIME タイプではない場合、PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.param_parse_err
mail_errors.bad_message_var
```

構文

```
PROCEDURE get_multipart_bodyparts (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
bodypart_list OUT MAIL_BODYPART_LIST);
PROCEDURE get_multipart_bodyparts (
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
bodypart_list OUT MAIL_BODYPART_LIST);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
bodypart_list	ボディパートのリスト

GET_MSG プロシージャ

このプロシージャは、エンコードされたメッセージ全体に対する BLOB ロケータを返します。事前に記憶域を割り当てる必要はありません。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE get_msg (
session_id     IN  NUMBER,
message_obj    IN  MAIL_MESSAGE_OBJ,
message_source OUT BLOB);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
message_source	元のエンコード形式でのメッセージ全体の内容

GET_MSG_BODY プロシージャ

このプロシージャは、メッセージ本体を指定された BLOB ロケータにコピーします。ロケータには、データを格納できる十分な容量が必要です。メッセージがシンプルな MIME タイプではない場合、データは返されません。メッセージ本体の Content-Transfer-Encoding ヘッダーで、base64 または quoted-printable エンコーディングを使用してデータをエンコードするように指定されている場合、内容はデコードしてから返されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
```

構文

```
PROCEDURE get_msg_body (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
content      OUT BLOB);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
content	デコードされたメッセージ全体の内容

GET_BODYPART_CONTENT プロシージャ

このプロシージャは、ボディパートの内容を指定された BLOB ロケータにコピーします。ボディパート・オブジェクトがシンプルな MIME タイプではない場合、データは返されません。ボディパートの Content-Transfer-Encoding ヘッダーで、base64 または quoted-printable エンコーディングを使用してデータをエンコードするように指定されている場合、内容はデコードしてから返されます。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.bad_message_var
```

構文

```
PROCEDURE get_bodypart_content (
session_id   IN  NUMBER,
bodypart_obj IN  MAIL_BODYPART_OBJ,
content      OUT BLOB);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
content	デコードされたメッセージ全体の内容

GET_MSGS_FLAGS プロシージャ

このプロシージャは、メッセージのフラグを返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE get_msgs_flags (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
message_flags OUT NUMBER);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
message_flags	リクエストされたメッセージのリストに対応するメッセージ・フラグのリスト。フラグの値は、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEEN_FLAG ■ MAIL_MESSAGE.GC_FLAGGED_FLAG ■ MAIL_MESSAGE.GC_ANSWERED_FLAG ■ MAIL_MESSAGE.GC_DELETED_FLAG ■ MAIL_MESSAGE.GC_DRAFT_FLAG

SET_MSGS_FLAGS プロシージャ

このプロシージャは、指定されたメッセージ・オブジェクトのメッセージ・フラグを設定および設定解除します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.folder_closed_err
```

構文

```
PROCEDURE set_msgs_flags (
session_id    IN  NUMBER,
message_obj   IN  MAIL_MESSAGE_OBJ,
message_flags IN  NUMBER,
set_flag      IN  BOOLEAN);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
message_flags	リクエストされたメッセージのリストに対応するメッセージ・フラグのリスト。フラグの値は、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEEN_FLAG ■ MAIL_MESSAGE.GC_FLAGGED_FLAG ■ MAIL_MESSAGE.GC_ANSWERED_FLAG ■ MAIL_MESSAGE.GC_DELETED_FLAG ■ MAIL_MESSAGE.GC_DRAFT_FLAG
set_flag	True の場合は指定されたフラグを設定し、False の場合は設定を解除します。

GET_AUTH_INFO プロシージャ

このプロシージャは、認証ユーザー情報を返します（ある場合）。認証ユーザー情報は、電子メール送信前のユーザー認証時に格納されます。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE get_auth_info (
session_id   IN  NUMBER,
message_obj  IN  MAIL_MESSAGE_OBJ,
auth_info    OUT VARCHAR2);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
auth_info	認証ユーザー情報

COMPOSE_MESSAGE プロシージャ

このプロシージャは、メッセージの作成を初期化します。一度に作成できるメッセージは 1 つです。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE compose_message (
message_obj   OUT MAIL_MESSAGE_OBJ);
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト

SET_MSGHEADER プロシージャ

このプロシージャは、一般的なメッセージ・ヘッダーのリストを設定します。NULL を指定すると、ヘッダーは含まれません。送信日が NULL の場合、現在の日付に設定されます。

発生する例外

mail_errors.msg_compose_limit_err

構文

```
PROCEDURE set_msgheader (
message_obj   IN  MAIL_MESSAGE_OBJ,
to_str        IN  VARCHAR2,
from_str      IN  VARCHAR2,
cc_str        IN  VARCHAR2 DEFAULT null,
replyto_str   IN  VARCHAR2 DEFAULT null,
sent_date     IN  DATE DEFAULT null,
subject_str   IN  VARCHAR2 DEFAULT null,
mime_version IN  VARCHAR2 DEFAULT '1.0',
content_type IN  VARCHAR2 DEFAULT 'text/plain',
charset       IN  VARCHAR2 DEFAULT 'us-ascii',
encoding      IN  VARCHAR2 DEFAULT '8bit');
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト
to_str	To RFC822 ヘッダー
from_str	From RFC822 ヘッダー
cc_str	Cc RFC822 ヘッダー
replyto_str	Reply-to RFC822 ヘッダー
sent_date	Date RFC822 ヘッダー
subject_str	Subject RFC822 ヘッダー

パラメータ	説明
mime_version	MIME-Version RFC822 ヘッダー
content_type	Content-Type RFC822 ヘッダー
charset	Content-Type RFC822 ヘッダーのキャラクタ・セット属性
encoding	Content-Transfer-Encodin RFC822 ヘッダー

SET_BPHEADER プロシージャ

このプロシージャは、一般的なボディパート・ヘッダーのリストを設定します。NULL を指定すると、ヘッダーは含まれません。すべてのヘッダー値の最大長は 2000 文字です。2000 文字を超えると、MAIL_ERRORS.PARAM_PARSE_ERR が発生します。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE set_bpheader (
bodypart_obj IN  MAIL_BODYPART_OBJ,
content_type IN  VARCHAR2 DEFAULT 'text/plain',
charset      IN  VARCHAR2 DEFAULT 'us-ascii',
encoding     IN  VARCHAR2 DEFAULT '8bit',
contentID    IN  VARCHAR2 DEFAULT null,
language     IN  VARCHAR2 DEFAULT null,
contentMD5   IN  VARCHAR2 DEFAULT null,
description  IN  VARCHAR2 DEFAULT null,
disposition  IN  VARCHAR2 DEFAULT 'inline',
filename     IN  VARCHAR2 DEFAULT null);
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト
content_type	Content-Type ヘッダー
charset	Content-Type ヘッダーのキャラクタ・セット属性
encoding	Content-Transfer-Encoding ヘッダー
contentID	Content-ID ヘッダー

パラメータ	説明
language	Content-Language ヘッダー
contentMD5	Content-MD5 ヘッダー
description	Content-Description ヘッダー
disposition	Content-Disposition ヘッダー
filename	Content-Disposition ヘッダーのファイル名属性

SET_HEADER プロシージャ

このプロシージャは、指定されたヘッダー・プロンプトのヘッダー値を設定します。前のヘッダーは上書きされず、新たなヘッダー値が前のヘッダーに追加されます。すべてのヘッダー値の最大長は 2000 文字です。2000 文字を超えると、MAIL_ERRORS.PARAM_PARSE_ERR が発生します。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE set_header (
message_obj    IN  MAIL_MESSAGE_OBJ,
header_prompt IN  VARCHAR2,
header_value   IN  VARCHAR2);
PROCEDURE set_header (
bodypart_obj   IN  MAIL_BODYPART_OBJ,
header_prompt IN  VARCHAR2,
header_value   IN  VARCHAR2);
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
header_prompt	メッセージ・ヘッダーまたはボディパート・ヘッダー
header_value	対応するヘッダー値

ADD_BODYPART プロシージャ

このプロシージャは、Content-Type にマルチパートが指定された親メッセージまたはボディパートに、子ボディパートを追加します。親メッセージ・オブジェクトまたはボディパート・オブジェクトが同じタイプのマルチパート・メッセージではない場合、PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE add_bodypart (
parent_message_obj  IN  MAIL_MESSAGE_OBJ,
bodypart_obj        OUT MAIL_BODYPART_OBJ);
PROCEDURE add_bodypart (
parent_bodypart_obj IN  MAIL_BODYPART_OBJ,
bodypart_obj        OUT MAIL_BODYPART_OBJ);
```

パラメータ

パラメータ	説明
parent_message_obj	親メッセージ・オブジェクト
parent_bodypart_obj	親ボディパート・オブジェクト
bodypart_obj	返される新しい子ボディパート・オブジェクト

ADD_INCLMSG_BODYPART プロシージャ

このプロシージャは、Content-Type にメッセージが指定された親メッセージまたはボディパートに、新しく含めるメッセージを追加します。親メッセージ・オブジェクトまたはボディパート・オブジェクトが同じタイプのメッセージではない場合、PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE add_inclmsg_bodypart (
parent_message_obj  IN  MAIL_MESSAGE_OBJ,
message_obj         OUT MAIL_MESSAGE_OBJ);
PROCEDURE add_inclmsg_bodypart (
parent_bodypart_obj IN  MAIL_BODYPART_OBJ,
message_obj         OUT MAIL_MESSAGE_OBJ);
```

パラメータ

パラメータ	説明
parent_message_obj	親メッセージ・オブジェクト
parent_bodypart_obj	親ボディパート・オブジェクト
message_obj	返される新しく含めるメッセージ・オブジェクト

SET_INCLMSG_BODYPART プロシージャ

このプロシージャは、Content-Type にメッセージが指定された親メッセージまたはボディパートに含めるメッセージを設定します。含めるメッセージが、メール・ストアに存在している必要があります。親メッセージ・オブジェクトまたはボディパート・オブジェクトが同じタイプのメッセージではない場合、PARAM_PARSE_ERR 例外が発生します。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.param_parse_err
```

構文

```
PROCEDURE set_inclmsg_bodypart (
parent_message_obj  IN  MAIL_MESSAGE_OBJ,
message_obj         IN  MAIL_MESSAGE_OBJ);
PROCEDURE set_inclmsg_bodypart (
parent_bodypart_obj IN  MAIL_BODYPART_OBJ,
message_obj         IN  MAIL_MESSAGE_OBJ);
```

パラメータ

パラメータ	説明
parent_message_obj	親メッセージ・オブジェクト
parent_bodypart_obj	親ボディパート・オブジェクト
message_obj	メール・ストア内の既存メッセージ

SET_CONTENT プロシージャ

このプロシージャは、作成中のメッセージにメッセージまたはボディパートの内容を設定します。メッセージまたはボディパートがシンプルな MIME タイプではない場合、PARAM_PARSE_ERR が発生します。このプロシージャは、何度でもコールできます。データは連結されます。データはデコードされている必要があります。作成したメッセージを送信または追加すると、データのそのパートに指定された Content-Transfer-Encoding ヘッダーに従ってエンコードされます。

発生する例外

mail_errors.msg_compose_limit_err
mail_errors.param_parse_err

構文

```
PROCEDURE set_content (
message_obj   IN  MAIL_MESSAGE_OBJ,
content       IN  RAW);
PROCEDURE set_content (
message_obj   IN  MAIL_MESSAGE_OBJ,
content       IN  BLOB);
PROCEDURE set_content (
bodypart_obj  IN  MAIL_BODYPART_OBJ,
content       IN  RAW);
PROCEDURE set_content (
bodypart_obj  IN  MAIL_BODYPART_OBJ,
content       IN  BLOB);
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・オブジェクト
content	メッセージまたはボディパートの内容

SEND_MESSAGE プロシージャ

このプロシージャは、現在作成しているメッセージを送信します。メッセージは、暗号化または署名付き、あるいはその両方で送信できます。

発生する例外

```
mail_errors.msg_compose_limit_err
mail_errors.smime_err
```

構文

```
PROCEDURE send_message (
message_obj       IN  MAIL_MESSAGE_OBJ);
PROCEDURE send_message (
message_obj       IN  MAIL_MESSAGE_OBJ,
ceritificate      IN  RAW,
private_key       IN  RAW,
recipients        IN  MAIL_MESSAGE.RAW_TABLE,
inclOrigCert      IN  BOOLEAN,
inclOrigAsRecip   IN  BOOLEAN,
digest_algorithm  IN  BINARY_INTEGER,
sign_algorithm    IN  BINARY_INTEGER,
encrypt_algorithm IN  BINARY_INTEGER,
send_option       IN  NUMBER);
```

パラメータ

パラメータ	説明
message_obj	メッセージ・オブジェクト。
certificate	ユーザーの証明書。
private_key	ユーザーの秘密鍵。
recipients	受信者の証明書。

パラメータ	説明
inclOrigCert	暗号化の際、署名の際、あるいはその両方でユーザーの証明書を含めるかどうかを指定します。
inclOrigAsRecip	受信者のリストにユーザーの証明書を含めるかどうかを指定します。
digest_algorithm	署名の際にダイジェストの生成に使用するアルゴリズム。メッセージの暗号化のみを行う場合は使用されません。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_MD5 ■ MAIL_MESSAGE.GC_SHA1
sign_algorithm	署名のアルゴリズム。メッセージの暗号化のみを行う場合は使用されません。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_RSA ■ MAIL_MESSAGE.GC_DSA
encrypt_algorithm	暗号化のアルゴリズム。メッセージに署名するだけの場合は使用されません。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_DES_EDE3_CBC ■ MAIL_MESSAGE.GC_DES_CBC ■ MAIL_MESSAGE.GC_RC2_CBC_128 ■ MAIL_MESSAGE.GC_RC2_CBC_40
send_option	メッセージを署名付きで、暗号化して、あるいはその両方で送信するかどうかを示します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEND_SIGNED ■ MAIL_MESSAGE.GC_ENCRYPTED ■ MAIL_MESSAGE.GC_SEND_SIGNED | MAIL_MESSAGE.GC_ENCRYPTED

APPEND_MESSAGE プロシージャ

このプロシージャは、現在作成しているメッセージを指定フォルダに追加します。ユーザーが認証され、フォルダがユーザーに属している必要があります。

発生する例外

```
mail_errors.unathenticated_err
mail_errors.msg_compose_limit_err
```

構文

```
PROCEDURE append_message (
session_id     IN  NUMBER,
message_obj    IN  MAIL_MESSAGE_OBJ,
folder_name    IN  VARCHAR2,
received_date  IN  DATE DEFAUL null,
message_flags  IN  NUMBER DEFAULT 0);
PROCEDURE append_message (
session_id     IN  NUMBER,
message_obj    IN  MAIL_MESSAGE_OBJ,
folder_obj     IN  MAIL_FOLDER_OBJ,
received_date  IN  DATE DEFAUL null,
message_flags  IN  NUMBER DEFAULT 0);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
folder_name	メッセージを追加するフォルダの名前。
folder_obj	メッセージを追加するフォルダのオブジェクト。
received_date	メッセージの受信日。

パラメータ	説明
message_flags	リクエストされたメッセージのリストに対応するメッセージ・フラグのリスト。フラグの値は、MAIL_MESSAGE パッケージ仕様で定義します。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_SEEN_FLAG ■ MAIL_MESSAGE.GC_FLAGGED_FLAG ■ MAIL_MESSAGE.GC_ANSWERED_FLAG ■ MAIL_MESSAGE.GC_DELETED_FLAG ■ MAIL_MESSAGE.GC_DRAFT_FLAG

DECRYPT_MESSAGE プロシージャ

このプロシージャは S/MIME で暗号化されているボディパートのリストを復号化して返します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.smime_err
```

構文

```
PROCEDURE decrypt_message (
session_id     IN  NUMBER,
message_obj    IN  MAIL_MESSAGE_OBJ,
ceritificate   IN  RAW,
private_key    IN  RAW,
bodypart_list  OUT MAIL_BODYPART_LIST
);
PROCEDURE decrypt_message (
session_id     IN  NUMBER,
bodypart_obj   IN  MAIL_MESSAGE_OBJ,
ceritificate   IN  RAW,
private_key    IN  RAW,
bodypart_list  OUT MAIL_BODYPART_LIST
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
message_obj	メッセージ・オブジェクト
bodypart_obj	ボディパート・メッセージ
certificate	ユーザーの証明書
private_key	ユーザーの秘密鍵
bodypart_list	復号化されたボディパートのリスト

VERIFY_MESSAGE プロシージャ

このプロシージャは、デジタル署名されたメッセージを確認します。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.smime_err
```

構文

```
PROCEDURE verify_message (
session_id       IN  NUMBER,
ceritificate     IN  RAW,
private_key      IN  RAW,
original_content IN  BLOB,
certificate_list IN  ES_CERT_LIST,
signature        IN  BLOB,
retruned_content OUT BLOB
);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
certificate	ユーザーの証明書
private_key	ユーザーの秘密鍵
original_content	メッセージの内容
certificate_list	証明書のリスト

パラメータ	説明
signature	デジタル署名

GET_THEMES プロシージャ

このプロシージャは、メッセージまたはボディパートのテーマを取得します。メッセージ・オブジェクトを指定すると、メッセージ全体が処理されます。ボディパート・オブジェクトを指定する場合、そのパートはシンプル・タイプで、他のボディパートが含まれていない必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.sql_err
mail_erros.imt_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
mail_errors.no_binary_err
```

構文

```
PROCEDURE get_themes (
session_id        IN  NUMBER,
message_obj       IN  MAIL_MESSAGE_OBJ,
flags             IN  INTEGER,
incl_binary_parts IN  BOOLEAN,
theme_buffer      OUT ES_OT_API.THEME_TABLE);
PROCEDURE get_themes (
session_id        IN  NUMBER,
bodypart_obj      IN  MAIL_BODYPART_OBJ,
flags             IN  INTEGER,
incl_binary_parts IN  BOOLEAN,
theme_buffer      OUT ES_OT_API.THEME_TABLE);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
bodypart_obj	ボディパート・オブジェクト。
flags	現在使用されません。
incl_binary_parts	False の場合、テキスト以外のパートは無視されます。

パラメータ	説明
theme_buffer	テーマ・バッファ。

GET_HIGHLIGHT プロシージャ

このプロシージャは、メッセージまたはボディパートのハイライトを取得します。メッセージ・オブジェクトを指定すると、メッセージ全体が処理されます。ボディパート・オブジェクトを指定する場合、そのパートはシンプル・タイプで、他のボディパートが含まれていない必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.sql_err
mail_erros.imt_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
mail_errors.no_binary_err
```

構文

```
PROCEDURE get_highlight (
session_id        IN  NUMBER,
message_obj       IN  MAIL_MESSAGE_OBJ,
flags             IN  INTEGER,
text_query        IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
highlight_buffer  OUT ES_OT_API.HIGHLIGHT_TABLE);
PROCEDURE get_highlight (
session_id        IN  NUMBER,
bodypart_obj      IN  MAIL_BODYPART_OBJ,
flags             IN  INTEGER,
text_query        IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
highlight_buffer  OUT ES_OT_API.HIGHLIGHT_TABLE);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
bodypart_obj	ボディパート・オブジェクト。

パラメータ	説明
flags	返されるバッファの書式。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_TEXT_FORMAT ■ MAIL_MESSAGE_GC_HTML_FORMAT
text_query	問い合せる文字列。
incl_binary_parts	False の場合、テキスト以外のパートは無視されます。
highlight_buffer	ハイライト・バッファ。

GET_MARKUPTEXT プロシージャ

このプロシージャは、メッセージまたはボディパートのマークアップ・テキストを取得します。メッセージ・オブジェクトを指定すると、メッセージ全体が処理されます。ボディパート・オブジェクトを指定する場合、そのパートはシンプル・タイプで、他のボディパートが含まれていない必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.sql_err
mail_erros.imt_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
mail_errors.no_markup
mail_errors.no_binary_err
```

構文

```
PROCEDURE get_markuptext (
session_id        IN  NUMBER,
message_obj       IN  MAIL_MESSAGE_OBJ,
flags             IN  INTEGER,
text_query        IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
tag_set           IN  VARCHAR2 DEFAULT 'TEXT_DEFUALT',
start_tag         IN  VARCHAR2 DEFAULT NULL,
end_tag           IN  VARCHAR2 DEFAULT NULL,
prev_tag          IN  VARCHAR2 DEFAULT NULL,
next_tag          IN  VARCHAR2 DEFAULT NULL,
buffer           OUT CLOB);
PROCEDURE get_markuptext (
session_id        IN  NUMBER,
bodypart_obj      IN  MAIL_BODYPART_OBJ,
```

```
flags             IN  INTEGER,
text_query        IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
tag_set           IN  VARCHAR2 DEFAULT 'TEXT_DEFUALT',
start_tag         IN  VARCHAR2 DEFAULT NULL,
end_tag           IN  VARCHAR2 DEFAULT NULL,
prev_tag          IN  VARCHAR2 DEFAULT NULL,
next_tag          IN  VARCHAR2 DEFAULT NULL,
buffer           OUT CLOB);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
bodypart_obj	ボディパート・オブジェクト。
flags	返されるバッファの書式。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_TEXT_FORMAT ■ MAIL_MESSAGE_GC_HTML_FORMAT
text_query	問い合せる文字列。
incl_binary_parts	False の場合、テキスト以外のパートは無視されます。
tag_set	Oracle Text のマニュアルを参照してください。
star_ttag	Oracle Text のマニュアルを参照してください。
end_tag	Oracle Text のマニュアルを参照してください。
prev_tag	Oracle Text のマニュアルを参照してください。
next_tag	Oracle Text のマニュアルを参照してください。
buffer	マークアップ・テキストのバッファ。

GET_FILTERED_TEXT プロシージャ

このプロシージャは、メッセージまたはボディパートのフィルタリングされたテキストを取得します。メッセージ・オブジェクトを指定すると、メッセージ全体が処理されます。ボディパート・オブジェクトを指定する場合、そのパートはシンプル・タイプで、他のボディパートが含まれていない必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.sql_err
mail_erros.imt_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
mail_errors.no_binary_err
```

構文

```
PROCEDURE get_filtered_text (
session_id        IN  NUMBER,
message_obj       IN  MAIL_MESSAGE_OBJ,
flags             IN  INTEGER,
incl_binary_parts IN  BOOLEAN,
buffer            OUT CLOB);
PROCEDURE get_filtered_text (
session_id        IN  NUMBER,
bodypart_obj      IN  MAIL_BODYPART_OBJ,
flags             IN  INTEGER,
incl_binary_parts IN  BOOLEAN,
buffer            OUT CLOB);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
bodypart_obj	ボディパート・オブジェクト。
flags	返されるバッファの書式。 値は次のとおりです。 ■ MAIL_MESSAGE.GC_TEXT_FORMAT ■ MAIL_MESSAGE_GC_HTML_FORMAT
incl_binary_parts	False の場合、テキスト以外のパートは無視されます。

パラメータ	説明
buffer	フィルタリングされたテキストのバッファ。

GET_TOKENS プロシージャ

このプロシージャは、メッセージまたはボディパートのトークンを取得します。メッセージ・オブジェクトを指定すると、メッセージ全体が処理されます。ボディパート・オブジェクトを指定する場合、そのパートはシンプル・タイプで、他のボディパートが含まれていない必要があります。

発生する例外

```
mail_errors.unauthenticated_err
mail_errors.sql_err
mail_erros.imt_err
mail_errors.bad_message_var
mail_errors.bad_msgpart_var
mail_errors.no_binary_err
```

構文

```
PROCEDURE get_tokens (
session_id        IN  NUMBER,
message_obj       IN  MAIL_MESSAGE_OBJ,
language          IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
token_buffer      OUT ES_OT_API.TOKEN_TABLE);
PROCEDURE get_tokens (
session_id        IN  NUMBER,
bodypart_obj      IN  MAIL_BODYPART_OBJ,
language          IN  VARCHAR2,
incl_binary_parts IN  BOOLEAN,
token_buffer      OUT ES_OT_API.TOKEN_TABLE);
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子。
message_obj	メッセージ・オブジェクト。
bodypart_obj	ボディパート・オブジェクト。
language	メッセージまたはボディパートのデータの言語。
incl_binary_parts	False の場合、テキスト以外のパートは無視されます。

パラメータ	説明
token_buffer	トークンのバッファ。

例外

ここでは、PL/SQL API の次の例外について説明します。

- external_rule_err 例外
- external_cond_err 例外
- too_many_rules 例外
- sql_err 例外
- imt_err 例外
- bad_message_var 例外
- bad_msgpart_var 例外
- no_binary_err 例外
- unauthenticated_err 例外
- folder_closed_err 例外
- msg_compose_limit_err 例外
- folder_not_found_err 例外
- folder_already_exists_err 例外
- operation_not_allowed 例外
- param_parse_err 例外
- internal_err 例外
- folder_name_err 例外
- login_err 例外
- folder_type_err 例外
- smime_err 例外

external_rule_err 例外

エラー番号 : 20001

メッセージ : Error executing external rule

原因 : 外部 PL/SQL プロシージャとして定義されたルールが実行中に失敗しました。

処置 : 外部 PL/SQL プロシージャが正しいかどうかをチェックしてください。

external_cond_err 例外

エラー番号 : 20002

メッセージ : External condition failed

原因 : 外部 PL/SQL ファンクションとして定義された条件が評価中に失敗しました。

処置 : 外部 PL/SQL ファンクションが正しいかどうかをチェックしてください。

too_many_rules 例外

エラー番号 : 20003

メッセージ : Too many rules fired, stop

原因 : メッセージに対してトリガーされたルールの数が、ルールの許容最大数を超えました。デフォルトは 20 です。

処置 :

- 使用したルールが無限ループの原因になっていないかどうかをチェックしてください。
- 定義するルールの数を減らしてください。
- ルールの許容最大数を増やすよう管理者に依頼してください。

sql_err 例外

エラー番号 : 20101

メッセージ : Some SQL error occured: %s

原因 : SQL エラーが発生しました。詳細は、返されたエラー・テキストを参照してください。

処置 : 『Oracle9*i* データベース・エラー・メッセージ』を参照してください。

imt_err 例外

エラー番号 : 20102

メッセージ : interMedia Text error: %s

原因 : 内部 Oracle Text エラーが発生しました。詳細は、返されたエラー・テキストを参照してください。

処置 : Oracle Text のマニュアルを参照してください。

bad_message_var 例外

エラー番号 : 20103

メッセージ : No such message: %s

原因 : 渡されたメッセージ・オブジェクトまたはボディパート・オブジェクトが無効な場合、例外が発生します。別のメール・セッションによってメッセージが削除された可能性があります。

処置 : 渡されたメッセージ・オブジェクトまたはボディパート・オブジェクトが有効であることを確認してください。

bad_msgpart_var 例外

エラー番号 : 20104

メッセージ : No such message part: %s

原因 : 渡されたメッセージ・オブジェクトまたはボディパート・オブジェクトの MIME レベルが無効な場合、例外が発生します。

処置 : 渡されたメッセージ・オブジェクトまたはボディパート・オブジェクトの MIME レベルが有効であることを確認してください。

no_binary_err 例外

エラー番号 : 20106

メッセージ : No binary part: %s

原因 : API オプションでバイナリが False のときに、指定されたメッセージ・パートがバイナリの場合、例外が発生します。

処置 : Oracle Text に対して、`withbinary` パラメータを `True` に設定してください。

unauthenticated_err 例外

エラー番号 : 20201

メッセージ : User needs to be authenticated first!

原因 : 現在ユーザーが認証されていません。

処置 : `mail_session.login()` プロシージャをコールして、ユーザーを認証してください。

folder_closed_err 例外

エラー番号 : 20202

メッセージ : Folder needs to be opened first!

原因 : 特定の操作においては、先に関連するフォルダを開く必要があります。

処置 : まず `mail_folder.open_folder()` プロシージャをコールしてフォルダを開いてください。

msg_compose_limit_err 例外

エラー番号 : 20203

メッセージ : Compose one message at a time!

原因 : ユーザーが一度に複数のメッセージを作成しようとすると、例外が発生します。

処置 : 2 番目のメッセージを作成する前に、現在のメッセージを送信または追加してください。

folder_not_found_err 例外

エラー番号 : 20204

メッセージ : Folder does not exist: %s

原因 : ユーザーがメール・ストアに存在しないフォルダで操作を実行しようとすると、例外が発生します。

処置 : 操作を実行する前にフォルダが存在することを確認してください。

folder_already_exists_err 例外

エラー番号 : 20205

メッセージ : Folder already exists: %s

原因 : ユーザーがすでに存在するフォルダ名を使用してフォルダの作成やフォルダ名の変更を試みると、例外が発生します。

処置 : 異なるフォルダ名を選択して再試行してください。

operation_not_allowed 例外

エラー番号 : 20206

メッセージ : Operation not allowed: %s

原因 : 次の原因が考えられます。

- `INBOX` フォルダを削除しようとしたこと。
- フォルダ名を、子の古い名前（x/y など）に変更しようとしたこと。

処置 : 名前の変更操作で、フォルダ階層を再編成しないでください。フォルダ階層を作成する場合は、作成および削除操作を使用してください。

param_parse_err 例外

エラー番号 : 20208

メッセージ : Param parsing error: %s

原因 : API に渡されたパラメータのエラー。次の原因が考えられます。

- 指定されたメッセージ UID が現在のフォルダに存在しないこと
- 現在作成していないメッセージを送信または追加しようとしていること
- 正しくない Content-Type 値を使用してメッセージ・オブジェクトまたはボディパート・オブジェクトを渡したこと
- 無効なソート条件
- 検索文字列でカッコや引用符が一致しないこと
- サポートされていない検索条件
- 不明な検索条件
- ヘッダー値が 2000 文字の制限より長いこと
- NULL フォルダ名でフォルダを作成しようとしていること

処置 : API に渡されたパラメータを修正し、再試行してください。

internal_err 例外

エラー番号 : 20209

メッセージ : Internal error: %s

原因 : 内部アサーションに失敗しました。データが整合していません。

処置 : オラクル社カスタマ・サポート・センターに連絡してください。

folder_name_err 例外

エラー番号 : 20210

メッセージ : Bad folder name: %s

原因 : 別のユーザーのネームスペースでフォルダの作成や名前の変更を試みています。

処置 : フォルダ名を訂正し、再試行してください。

login_err 例外

エラー番号 : 20211

メッセージ : Oracle Internet Directory Login Error: %s

原因 : 無効なユーザー名またはパスワードを指定すると、例外が発生します。

処置 : スペルをチェックし、再試行してください。

folder_type_err 例外

エラー番号 : 20212

メッセージ : Folder type violation: %s

原因 : 次の原因が考えられます。

- 選択できないフォルダを開こうとしていること
- サブフォルダを作成できない親フォルダでフォルダを作成しようとしていること
- サブフォルダを持つ選択できないフォルダを削除しようとしていること
- 選択できないフォルダにメッセージをコピーしようとしていること

処置 : これらのタイプのフォルダ違反がないようにしてください。

smime_err 例外

エラー番号 : 20213

メッセージ : S/MIME error: %s

原因 : S/MIME ファンクションのコールでエラーが発生しました。詳細は、S/MIME エラー・コードを参照してください。

処置 : エラー・コードの説明については、S/MIME のマニュアルを参照してください。

例

この項では、ログイン方法の例、およびフォルダでの作成、リスト表示、検索、フェッチ、送信機能の使用方法の例を示します。GetTheme API の使用方法の例も示します。

この項では、次の項目について説明します。

- ログイン、作成、リスト表示および検索の例
- ログインおよびすべてフェッチの例
- 作成および送信の例
- GetTheme の例

ログイン、作成、リスト表示および検索の例

この例では、フォルダ上でログイン、作成、リスト表示および検索を行う方法を示します。

```
set serveroutput on size 1000000;

DECLARE
   user_name            VARCHAR2(50) := 'testuser1';
   user_pswd            VARCHAR2(50):= 'welcome';
   user_domain          VARCHAR2(50) := 'oracle.com';
   ldap_server          VARCHAR2(50) := 'test-sun.us.oracle.com';
   ldap_port            NUMBER := 389;

   sessionID            NUMBER;
   my_folder_obj        MAIL_FOLDER_OBJ;
   my_folder_list       MAIL_FOLDER_LIST;
   inbox_obj            MAIL_FOLDER_OBJ;
   search_string        VARCHAR2(500);
   message_list         MAIL_MESSAGE_LIST;

   -- declare and define two procedures to recursively print out all folders
   PROCEDURE print_folder(session_id     IN  NUMBER,
                          folder_obj     IN  MAIL_FOLDER_OBJ);
   PROCEDURE print_folderlist(session_id IN  NUMBER,
                              flist      IN  MAIL_FOLDER_LIST);

   PROCEDURE print_folder(session_id     IN  NUMBER,
                          folder_obj     IN  MAIL_FOLDER_OBJ) IS
        flist   MAIL_FOLDER_LIST;
   BEGIN
      -- print out the folder name
      dbms_output.put_line(folder_obj.name);
```

```
      -- if there are child-folder, print them too
      if mail_folder.has_folder_children(session_id, folder_obj) then
         -- get all the child-folders and prin them
         mail_folder.list_folders(session_id, folder_obj, flist);
         print_folderlist(flist);
      end if;
   END;

   PROCEDURE print_folderlist(session_id IN  NUMBER,
                              flist      IN  MAIL_FOLDER_LIST) IS
   BEGIN
      for i in 1..flist.count loop
         print_folder(session_id, flist(i));
      end loop;
   END;

BEGIN
   -- login
   mail_session.login(user_name, user_pswd, user_domain,
                      ldap_server, sessionID, ldap_port);
   dbms_output.put_line(user_name || ' logged-in');

   -- create a new folder called my_folder
   mail_folder.create_folder(sessionID, 'my_folder', my_folder_obj);
   dbms_output.put_line('Created my_folder folder');

   -- list all folders
   mail_folder.list_toplevel_folders(sessionID, my_folder_list);
   dbms_output.put_line('My folder list:');
   print_folderlist(sessionID, my_folder_list);

   -- open INBOX
   mail_folder.open_folder(sessionID, 'INBOX', inbox_obj);
   dbms_output.put_line('Opened INBOX');

   -- search on INBOX for mails from Crosby with subject containing the word
   -- White Christmas
   search_string := 'from Crosby subject White Christmas';
   mail_folder.search_folder(sessionID, inbox_obj, search_string, message_list);
   dbms_output.put_line('Search for ' || search_string || ' returns:');
   FOR i IN 1..message_list.COUNT LOOP
      dbms_output.put(message_list(i).msg_uid || ' ');
   END LOOP;
   dbms_output.new_line;
```

```
   -- logout
   mail_session.logout(sessionID);
   dbms_output.put_line('Elvis has left the building!');

   -- commit
   commit;
EXCEPTION
   WHEN mail_errors.login_err THEN
      dbms_output.put_line('Failed to log-in!!!');
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
   WHEN mail_errors.folder_already_exists_err OR
        mail_errors.folder_type_err THEN
      dbms_output.put_line('Failed to create folder!!!');
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
   WHEN OTHERS THEN
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
END;
/
```

ログインおよびすべてフェッチの例

この例では、ログインしてフォルダからすべてのメッセージをフェッチする方法を示します。

```
set serveroutput on size 1000000;

DECLARE
   user_name            VARCHAR2(50) := 'testuser1';
   user_pswd            VARCHAR2(50):= 'welcome';
   user_domain          VARCHAR2(50) := 'oracle.com';
   ldap_server          VARCHAR2(50) := 'test-sun.us.oracle.com';
   ldap_port            NUMBER := 389;

   sessionID            NUMBER;
   inbox_obj            MAIL_FOLDER_OBJ;
   message_list         MAIL_MESSAGE_LIST;
```

```
-- declare and define some sub-programs to print out a message
PROCEDURE print_message (sessionId   IN  NUMBER,
                         p_msg_obj   IN  MAIL_MESSAGE_OBJ);

PROCEDURE print_bodypart (sessionId  IN  NUMBER,
                          p_bp_obj   IN  MAIL_BODYPART_OBJ);

PROCEDURE print_header (sessionId    IN  NUMBER,
                        p_msg_obj    IN  MAIL_MESSAGE_OBJ);

PROCEDURE print_header (sessionId    IN  NUMBER,
                        p_bp_obj     IN  MAIL_BODYPART_OBJ);

PROCEDURE print_content (sessionId           IN  NUMBER,
                         p_msg_obj           IN  MAIL_MESSAGE_OBJ,
                         p_content_type      IN  varchar2);

PROCEDURE print_content (sessionId           IN  NUMBER,
                         p_bp_obj            IN  MAIL_BODYPART_OBJ,
                         p_content_type      IN  varchar2);

PROCEDURE print_message (sessionId   IN  NUMBER,
                         p_msg_obj   IN  MAIL_MESSAGE_OBJ) IS
   bp_list           MAIL_BODYPART_LIST;
   content_type      varchar2(500);
   incl_msg          MAIL_MESSAGE_OBJ;
BEGIN
   -- print out the message header
   print_header(sessionId, p_msg_obj);

   -- get the message's content-type
   mail_message.get_content_type(sessionId, p_msg_obj, content_type);
   content_type := upper(content_type);

   -- if it is a multipart type, get each of its body-parts and print
   -- it out one by one
   if content_type like 'MULTIPART/%' then
      mail_message.get_multipart_bodyparts(sessionId, p_msg_obj, bp_list);
      for i in 1..bp_list.count loop
         print_bodypart(sessionId, bp_list(i));
      end loop;
```

```
   -- if it is a message type, get the included message and print
   -- the included message
   elsif content_type like 'MESSAGE/%' then
      mail_message.get_included_message(sessionId, p_msg_obj, incl_msg);
      print_message(sessionId, incl_msg);

   -- if it is a simple type, print the content
   else
      print_content(sessionId, p_msg_obj, content_type);
   end if;
END;

PROCEDURE print_bodypart (sessionId  IN  NUMBER,
                          p_bp_obj   IN  MAIL_BODYPART_OBJ) IS
   bp_list           MAIL_BODYPART_LIST;
   content_type      varchar2(500);
   incl_msg          MAIL_MESSAGE_OBJ;
BEGIN
   -- print out the body-part header
   print_header(sessionId, p_bp_obj);

   -- get the body-part's content-type
   mail_message.get_content_type(sessionId, p_bp_obj, content_type);
   content_type := upper(content_type);

   -- if it is a multipart type, get each of its body-parts and print
   -- it out one by one
   if content_type like 'MULTIPART/%' then
      mail_message.get_multipart_bodyparts(sessionId, p_bp_obj, bp_list);
      for i in 1..bp_list.count loop
         print_bodypart(sessionId, bp_list(i));
      end loop;

   -- if it is a message type, get the included message and print
   -- the included message
   elsif content_type like 'MESSAGE/%' then
      mail_message.get_included_message(sessionId, p_bp_obj, incl_msg);
      print_message(sessionId, incl_msg);

   -- if it is a simple type, print the content
   else
      print_content(sessionId, p_bp_obj, content_type);
   end if;
END;
```

```
--
-- private procedure to print header given a MAIL_HEADER_LIST object
--
PROCEDURE print_hdrlist (hdr_list    IN  MAIL_HEADER_LIST)
BEGIN
   dbms_output.put_line('[# message hdr: ' || hdr_list.count || ']');
   for i in 1..hdr_list.count loop
      dbms_output.put(hdr_list(i).header_prompt || ': ');
      for j in 1..hdr_list(i).header_value.count loop
         dbms_output.put_line(hdr_list(i).header_value(j));
      end loop;
   end loop;
END;

PROCEDURE print_header (sessionId    IN  NUMBER,
                        p_msg_obj    IN  MAIL_MESSAGE_OBJ) IS
   hdr_list          MAIL_HEADER_LIST;
   msg_size          NUMBER;
   msg_flags         NUMBER;
   msg_rcvd_date     VARCHAR2(50);
BEGIN
   if p_msg_obj.mime_level = '0' then        -- top level message
      mail_message.get_msg_size(sessionId, p_msg_obj, msg_size);
      mail_message.get_msg_flags(sessionId, p_msg_obj, msg_flags);
      mail_message.get_received_date(sessionId, p_msg_obj,
                                     'DD-Mon-YYYY HH24:MI:SS',
                                     msg_rcvd_date);

      dbms_output.put_line('[MSG_SIZE:' || msg_size || ']' ||
                           ' [MSG_FLAG:' || msg_flags || ']');
      dbms_output.put_line('[message header]:');
   else
      dbms_output.put_line('[included message header]:');
   end if;

   -- get all the headers for this message and print them
   mail_message.get_headers(sessionId, p_msg_obj, hdr_list);
   print_hdrlist(hdr_list);

   if p_msg_obj.mime_level = '0' then        -- top level message
     dbms_output.put_line('Received: '|| msg_rcvd_date);
   end if;

   dbms_output.new_line;
END;
```

```
PROCEDURE print_header (sessionId    IN  NUMBER,
                        p_bp_obj     IN  MAIL_BODYPART_OBJ) IS
   hdr_list          MAIL_HEADER_LIST;
BEGIN
   -- get all the headers for this body-part and print them
   dbms_output.put_line('[bodypart header]:');
   mail_message.get_headers(sessionId, p_bp_obj, hdr_list);
   print_hdrlist(hdr_list);
   dbms_output.new_line;
END;

PROCEDURE print_content (sessionId           IN  NUMBER,
                         p_msg_obj           IN  MAIL_MESSAGE_OBJ,
                         p_content_type      IN  varchar2) IS
   msg_data          BLOB;
   l_data            RAW(255);
   l_length          NUMBER;
   l_offset          NUMBER;
   l_size            NUMBER;
BEGIN
   -- create a temporary lob to hold the message body
   dbms_lob.createtemporary(msg_data, TRUE, dbms_lob.session);

   -- get the decoded message body
   mail_message.get_msg_body(sessionId, p_msg_obj, msg_data);
   l_length := dbms_lob.getlength(msg_data);
   dbms_output.put_line('[read ' || l_length || ']');
   dbms_output.new_line;

   -- print the decoded message body if it is a text type
   if p_content_type like 'TEXT/%' then
      l_offset := 1;
      while l_offset <= l_length loop
         -- sqlplus has a limitation of 255 characters per line.
         -- break the content into 250-character chunks and print.
         if (l_length - l_offset + 1) <= 250 then
            l_size := l_length - l_offset + 1;
         else
            l_size := 250;
         end if;
         l_data := dbms_lob.substr(msg_data, l_size, l_offset);
         l_offset := l_offset + l_size;
         dbms_output.put_line(utl_raw.cast_to_varchar2(l_data));
      end loop;
   end if;
```

```
   -- free the temporary lob allocated
   dbms_lob.freetemporary(msg_data);
END;

PROCEDURE print_content (sessionId           IN  NUMBER,
                         p_bp_obj            IN  MAIL_BODYPART_OBJ,
                         p_content_type      IN  varchar2) IS
   bp_data           BLOB;
   l_data            RAW(255);
   l_length          NUMBER;
   l_offset          NUMBER;
   l_size            NUMBER;
BEGIN
   -- create a temporary lob to hold the body-part content
   dbms_lob.createtemporary(bp_data, TRUE, dbms_lob.session);

   -- get the decoded body-part content
   mail_message.get_bodypart_content(sessionId, p_bp_obj, bp_data);
   l_length := dbms_lob.getlength(bp_data);
   dbms_output.put_line('[read ' || l_length || ']');
   dbms_output.new_line;

   -- print the decoded data if it is a text type
   if p_content_type like 'TEXT/%' then
      l_offset := 1;
      while l_offset <= l_length loop
         -- sqlplus has a limitation of 255 characters per line.
         -- break the content into 250-character chunks and print.
         if (l_length - l_offset + 1) <= 250 then
            l_size := l_length - l_offset + 1;
         else
            l_size := 250;
         end if;
         l_data := dbms_lob.substr(bp_data, l_size, l_offset);
         l_offset := l_offset + l_size;
         dbms_output.put_line(utl_raw.cast_to_varchar2(l_data));
      end loop;
   end if;

   -- free the temporary lob allocated
   dbms_lob.freetemporary(bp_data);
END;
```

```
   -- define a bit_on function
   FUNCTION bit_on (
        p_flag  IN  INTEGER,
        p_bit   IN  INTEGER
   ) RETURN INTEGER IS
   BEGIN
      IF bitand(p_flag, p_bit) = 0 THEN         -- bit not turned on yet
         RETURN p_flag + p_bit;
      ELSE                                      -- bit already on
         RETURN p_flag;
      END IF;
   END;


BEGIN
   -- login
   mail_session.login(user_name, user_pswd, user_domain,
                      ldap_server, sessionID, ldap_port);
   dbms_output.put_line(user_name || ' logged-in');

   -- open INBOX
   mail_folder.open_folder(sessionID, 'INBOX', inbox_obj);
   dbms_output.put_line('Opened INBOX');

   -- get all the messages from INBOX
   mail_folder.get_folder_messages(sessionId, message_list);
   dbms_output.put_line('Got ' || message_list.COUNT || ' messages');
   FOR i IN 1..message_list.COUNT LOOP
      dbms_output.put_line('***** Message # ' || i || ' *****');
      print_message(sessionId, message_list(i));

   END LOOP;

   -- set all the messages flags to be seen and deleted
   mail_folder.set_msg_flags(sessionId, message_list,
             bit_on(MAIL_MESSAGE.GC_SEEN_FLAG, MAIL_MESSAGE.GC_DELETED_FLAG),
             true);

   -- expunge INBOX to remove all the messages we have fetched
   mail_folder.expunge_folder(sessionId);

   -- logout
   mail_session.logout(sessionID);
   dbms_output.put_line('Elvis has left the building!');
```

```
   -- commit
   commit;
EXCEPTION
   WHEN mail_errors.login_err THEN
      dbms_output.put_line('Failed to log-in!!!');
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
   WHEN OTHERS THEN
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
END;
/
```

作成および送信の例

この例では、マルチパート / 代替メッセージの作成および送信方法を示します。

```
set serveroutput on size 1000000;

DECLARE
   to_addr              VARCHAR2(100) := 'testuser2@oracle.com';
   from_addr            VARCHAR2(100) := 'testuser1@oracle.com';
   subject_str          VARCHAR2(100) :=
        'Example: send a multipart/alternative message';

   msg_obj              mail_message_obj;
   a_bodypart           mail_bodypart_obj;
   b_bodypart           mail_bodypart_obj;
   msg_data             VARCHAR2(500);

BEGIN
   -- start composing a message
   mail_message.compose_message(msg_obj);
   dbms_output.put_line('start composing...');

   -- set the message's headers
   mail_message.set_msgheader(
        msg_obj, to_addr, from_addr, null, from_addr, sysdate,
        subject_str, '1.0', 'multipart/alternative', null, null
   );
   dbms_output.put_line('set message header');
```

```
   -- add the text/plain body-part
   mail_message.add_bodypart(msg_obj, a_bodypart);
   mail_message.set_bpheader(
        a_bodypart, 'text/plain', 'us-ascii', 'quoted-printable',
        null, null, null, null, null, null
   );
   msg_data := 'text/plain data';
   mail_message.set_content(a_bodypart, utl_raw.cast_to_raw(msg_data));
   dbms_output.put_line('set text/plain body');

   -- add the text/html body-part
   mail_message.add_bodypart(msg_obj, b_bodypart);
   mail_message.set_bpheader(
        b_bodypart, 'text/html', 'us-ascii', 'quoted-printable',
        null, null, null, null, null, null
   );
   msg_data := '<html>text/html data</html>';
   mail_message.set_content(b_bodypart, utl_raw.cast_to_raw(msg_data));
   dbms_output.put_line('set text/html body');

   -- now send this message
   mail_message.send_message(msg_obj);
   dbms_output.put_line('send message');

   -- commit
   commit;
EXCEPTION
   when OTHERS then
      dbms_output.put_line ('send failed!');
      dbms_output.put_line('SQLcode: ' || sqlcode);
      dbms_output.put_line('SQLerrm: ' || sqlerrm);
      rollback;
END;
/
```

GetTheme の例

この例では、GetTheme API の使用方法を示します。

```
ACCEPT msgid NUMBER DEFAULT 1 PROMPT 'Input message id: ';

set serveroutput on
DECLARE
  themebuf CTXSYS.CTX_DOC.THEME_TAB;
  errmsg   VARCHAR2(254);
  res      INTEGER;
  i        INTEGER;
BEGIN

  res := ES_OT_API.GetThemes(&msgid, '0' , 1,false, themebuf, errmsg);
  IF res != ES_OT_API.ESOTAPI_OK THEN
     DBMS_OUTPUT.PUT_LINE('GetTheme error no: ' || res);
     DBMS_OUTPUT.PUT_LINE(errmsg);
  ELSE
     dbms_output.put_line('found '|| themebuf.count ||' themes');
     FOR i in 1..themebuf.count LOOP
       DBMS_OUTPUT.PUT_LINE(' (' || themebuf(i).theme || ', ' ||
                            themebuf(i).weight || ') ');
     END LOOP;
     DBMS_OUTPUT.PUT_LINE('Get ' || themebuf.count || ' themes succeed');
  END IF;

EXCEPTION
  WHEN no_data_found THEN
     DBMS_OUTPUT.PUT_LINE('ERROR:' || errmsg);
END;
/
```

ABORT_MESSAGE プロシージャ

このプロシージャは、新規メッセージを作成および送信中のエラーの後で、クリーンアップを実行します。

構文

```
PROCEDURE abort_message;
```

GET_BP_CONTENT_WITH_HDRS プロシージャ

このプロシージャは、エンコードされたボディパートの内容をパート・ヘッダーとともに、指定された BLOB ロケータにコピーします。ロケータには、データを格納できる十分な容量が必要です。

発生する例外

```
mail_errors.unauthenticated_err
```

構文

```
PROCEDURE get_bp_content_with_hdrs (
session_id    IN  NUMBER,
bodypart_obj  IN  MAIL_BODYPART_OBJ,
content       IN OUT BLOB
) RETURN boolean;
```

パラメータ

パラメータ	説明
session_id	ユーザーの認証済セッションを表す識別子
bodypart_obj	ボディパート・オブジェクト
content	ヘッダー付きのエンコードされたボディパートの内容

2

Java API リファレンス

この章では、カスタマイズされたクライアントの作成、アプリケーションの統合、特定の拡張機能のサポートに使用する Oracle Email Java API について説明します。

次の項目について説明します。

- JavaMail API
- ディレクトリ管理 API
- ルール管理 API
- 携帯情報端末用フィルタおよびプロファイル API

> **関連資料：** Java API の詳細は、「Oracle Email API Reference」（Java Doc）を参照してください。

JavaMail API

Oracle Javamail API (OJMA) Service Provider は、Sun 社が提供する抽象 JavaMail™ 1.2 API（JMA）を実装し、Oracle データベースでメール・ストアと直接統合されるように設計されています。この手法では、標準ベースのプロトコル・サーバーに依存せず、メール・ストアで JDBC を使用して PL/SQL API をコールします。

JMA は、メール・システムをモデル化する抽象クラスのセットを提供します。API は、Java テクノロジに基づくメールおよびメッセージ・アプリケーションを構築するための、プラットフォームやプロトコルに依存しないフレームワークを提供します。JMA 実装は、一般に、IMAP や POP3 などの標準ベースのプロトコルに基づきます。JMA は Java プラットフォームのオプション・パッケージとして実装され、Java 2 Platform Enterprise Edition の一部として入手することもできます。詳細は、次の URL を参照してください。

```
http://java.sun.com/j2ee/ja/javamail/index.html
```

JMA 内では、ユーザーはメッセージ・ストアに接続します。ストアにはフォルダのリストが含まれ、ユーザーは IMAP プロトコル（RFC 2060）に基づいてフォルダおよびメッセージ操作を実行できます。

IMAP プロトコル・サービス・プロバイダを介した JMA 操作で使用可能な標準機能に加えて、Oracle Javamail API Service Provider は、IMAP プロトコルおよび JMA に対するいくつかの拡張機能をサポートしています。これらの拡張機能は基本的な JMA インタフェースに干渉しないため、基本的な JMA を使用するクライアントは、Oracle Javamail API Service Provider とシームレスに統合できます。

Oracle Javamail API Service Provider は次の拡張機能をサポートします。

- RFC2086 に基づく ACL 拡張子のサポート。メール・ユーザーは、自分のフォルダを他のユーザー、グループ、リストと共有できます。
- 管理者は、ドメイン・レベルでパブリック・フォルダを作成し、そのドメインのすべてのユーザーに対して使用可能にできます。
- IMAP Sort Extension ドラフトに基づくフォルダ内のメッセージのソート。
- SMIME ツールキットとの統合。送信電子メールを暗号化し、受信電子メールを復号化および確認できます。
- Oracle Text を使用したメッセージの内容に基づく検索。
- 携帯情報端末プロファイルおよびフィルタ。ストアでフィルタを適用し、フィルタリングされたメッセージのサブセットのみを表示できます（PDA や携帯情報端末からの電子メール・アクセスに有用）。
- IMAP Annotate Extension ドラフトに基づくメッセージ注釈。ユーザーがフォルダ内のメッセージにコメントや注釈を追加できます。

また、Oracle Javamail API Service Provider は Oracle データベースと直接連動するため、IMAP や POP3 などの標準ベース・サーバーとの対話によるオーバーヘッドがありません。そのためシステム要件が低減され、ソート、参照、メッセージ共有などのタスクを Oracle データベースで効率的に処理できます。

関連資料：

- OJMA の拡張機能の詳細は、「Oracle Email API Reference」（Java Doc）を参照してください。
- Javamail のドキュメントは、Javamail に関する Web サイトを参照してください。

この項では、次の JavaMail API の使用例を示します。

ユーザーのメッセージを開く

次の例では、特定のユーザーの受信ボックスにあるすべてのメッセージを開く方法を示します。

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;
import oracle.mail.ldap.ESDSConstants;
import oracle.mail.sdk.esmail.OracleAuthenticator;

public class ReadMessage
{
        static String password = null;
        static String user = null;
        static String ldapHost = null;
        static int ldapPort = -1;
        static String mbox = "INBOX";

        public static void main (String argv[]) throws Exception
        {
                for (int i = 0; i < argv.length; i++)
                {
```

```
        if (argv[i].equals("-U"))
                user = argv[++i];
        else if (argv[i].equals("-P"))
                password = argv[++i];
        else if (argv[i].equals("-D"))
                ldapHost = argv[++i];
        else if (argv[i].equals("-I"))
                ldapPort = Integer.parseInt(argv[++i]);
        else if (argv[i].equals("--")) {
                i++;
                break;
        }
        else if (argv[i].startsWith("-")) {
                System.out.println(
                "Usage: ReadMessage [-D ldapHost] [-I ldapPort]
                [-U user] [-P password]");
                System.exit(1);
        }
        else {
                break;
        }
}


Properties props = System.getProperties();

// Set system properties - it is necessary to set up these
// properties to obtain the database connect information
// from oid. The database connect information is available only
// to privileged users.
// NOTE: The password may be different
// for your installation.
props.setProperty("oracle.mail.ldap.admin_dn","cn=umadmin,
cn=emailservercontainer,cn=products,cn=oraclecontext");
props.setProperty("oracle.mail.ldap.admin_password","umadmin");

Session session = Session.getDefaultInstance(props,new
OracleAuthenticator());
session.setDebug(true);

Store store = null;

store = session.getStore("esmail");

if (user != null && password != null && ldapPort != 0 && ldapHost
!= null){
store.connect(ldapHost, ldapPort, user, password);
```

```
// Open the folder - after store has connected
Folder folder = store.getDefaultFolder();

folder = folder.getFolder(mbox);

if (folder == null) {
        System.out.println("Invalid folder");
        System.exit(1);
}
else {
        System.out.println("Folder name : " + folder.getName());
}

// try to open read/write and if that fails try read-only
try {
   folder.open(Folder.READ_WRITE);
} catch (MessagingException ex) {
   folder.open(Folder.READ_ONLY);
}

int totalMessages = folder.getMessageCount();

if (totalMessages == 0) {
   System.out.println("Empty folder");
   folder.close(false);
   store.close();
   System.exit(1);
}
else
{
   System.out.println("Total messages: " + totalMessages);

   Message[] msgs = folder.getMessages();
   if (msgs != null)
        System.out.println("ReadMessage: Number of messages: " +
        msgs.length);
   int my_uid = 0;
   for (int i = 0; i < msgs.length; i++)
   {
        System.out.println("----------------------------");
        System.out.println("This is the message uid# : " +
        (int)msgs[i].getMessageNumber());
        my_uid = msgs[i].getMessageNumber();
```

```
            if (my_uid > 0)
            {
               Message msg = folder.getMessage(my_uid);
               System.out.println("Sent Date: " + msg.getSentDate());
               System.out.println("Msg Size: " + msg.getSize());
               System.out.println("Received Date: " +
               msg.getReceivedDate());
               dumpPart(msg);
            }
            else
            {
               System.out.println("No messages returned");
            }
         }  //end for


    }
    }

}

public static void dumpPart(Part p) throws Exception
{
    if (p instanceof Message)
       dumpEnvelope((Message)p);

    System.out.println("----------------------------");
    pr("CONTENT-TYPE: " + p.getContentType());

    if (p.isMimeType("text/plain")) {
            pr("This is plain text");
            pr("---------------------------");
            System.out.println((String)p.getContent());
    } else if (p.isMimeType("multipart/*")) {
            pr("This is a Multipart");
            pr("---------------------------");
            Multipart mp = (Multipart)p.getContent();
            int count = mp.getCount();
            for (int i = 0; i < count; i++)
                  dumpPart(mp.getBodyPart(i));
    } else if (p.isMimeType("message/rfc822")) {
            pr("This is a Nested Message");
            pr("---------------------------");
            dumpPart((Part)p.getContent());
    } else if (p.isMimeType("image/jpeg")) {
        pr("--------> image/jpeg");
        Object o = p.getContent();
```

```
            if (o instanceof InputStream) {
            }
            InputStream x = (InputStream)o;
            // Construct the required byte array
            System.out.println("x.length = " + x.available());
            int i = 0;
            byte[] bArray = new byte[x.available()];

            while ((i = (int)((InputStream)x).available()) > 0)
            {
                int result = (int)(((InputStream)x).read(bArray));
                if (result == -1) break;
            }
            FileOutputStream f2 = new FileOutputStream("/tmp/image.jpg");
            f2.write(bArray);
        }
        else {
                Object o = p.getContent();
                if (o instanceof String) {
                        pr("This is a string");
                        pr("---------------------------");
                        System.out.println((String)o);
                } else if (o instanceof InputStream) {
                        pr("This is just an input stream");
                        pr("---------------------------");
                        InputStream is = (InputStream)o;
                        is = (InputStream)o;
                        int c;
                        while ((c = is.read()) != -1)
                             System.out.write(c);
                } else {
                        pr("This is an unknown type");
                        pr("---------------------------");
                        pr(o.toString());
                }
        }

    }

    public static void dumpEnvelope(Message m) throws Exception {
        pr("This is the message envelope");
        pr("---------------------------");
        Address[] a;
```

```
            // FROM
            if ((a = m.getFrom()) != null) {
                    for (int j = 0; j < a.length; j++)
                            pr("FROM: " + a[j].toString());
            }

            // TO
            if ((a = m.getRecipients(Message.RecipientType.TO)) != null) {
                    for (int j = 0; j < a.length; j++)
                            pr("TO: " + a[j].toString());
            }

            // SUBJECT
            if (m.getSubject() != null)
                    pr("SUBJECT: " + m.getSubject());

            // DATE
            Date d = m.getSentDate();
            pr("SendDate: " + (d != null ? d.toString() : "UNKNOWN"));
        }

        public static void pr(String s){
            System.out.println(s);
        }
}
```

共有フォルダの作成およびユーザー権限の付与

次の例では、共有フォルダを作成し、ユーザーに権限を付与する方法を示します。共有フォルダ名とユーザー（対象者）は、共有フォルダの所有者および所有者パスワードに加えてパラメータとして使用されます。

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;
import oracle.mail.sdk.esmail.*;

public class SharedFolderCreate
{
        static String password = null;
        static String user = null;
        static String aSharedFolderUser = null;
        static String aSharedFolderName = null;
        static String ldapHost = null;
        static int ldapPort = -1;
        static String mbox = "INBOX";
```

```
public static void main (String argv[]) throws Exception
{
        for (int i = 0; i < argv.length; i++)
        {
                if (argv[i].equals("-U"))
                        user = argv[++i];
                else if (argv[i].equals("-P"))
                        password = argv[++i];
                else if (argv[i].equals("-S"))
                        aSharedFolderUser = argv[++i];
                else if (argv[i].equals("-N"))
                        aSharedFolderName = argv[++i];
                else if (argv[i].equals("--")) {
                        i++;
                        break;
                }
                else if (argv[i].startsWith("-")) {
                        System.out.println(
                        "Usage: SharedFolderCreate [-U user] [-P password]
                        [-S sharedFolderUser] [-N sharedFolderName]");
                        System.exit(1);
                }
                else {
                        break;
                }
        }


        // This property is used to set the default oracle home
        // in the iasv2 environment. The default ldap host and
        // ldap port information is picked up from the
        // $ORACLE_HOME/config/ias.properties config file
        // In the store.connect method, the ldapHost is set to
        // null and the ldapPort is set to -1 to make sure that
        // the default values are used. In this example, the oracle
        // home is retrieved from a system level ORACLE_HOME
        // property.
        Properties props = System.getProperties();
        String orclHome = System.getProperty("ORACLE_HOME");
        props.setProperty("oracle.mail.ldap.oracle_home",orclHome);

        Session session = Session.getDefaultInstance(props,null);
        session.setDebug(true);

        Store store = null;
```

```
                    store = session.getStore("esmail");

                    if (user != null && password != null){
                       store.connect(null, -1, user, password);

                       // Open the folder - after store has connected
                       Folder folder = store.getFolder(aSharedFolderName);

                       if (!folder.exists()) {
                            System.out.println("Folder does not exist");
                            folder.create(Folder.HOLDS_MESSAGES);
                       }
                       System.out.println("Folder name : " + folder.getName());
                       System.out.println("Creating Shared Folder");
                       ((OracleFolder)folder).addACI(aSharedFolderUser,
                       OracleACI.READACI);
                       //((OracleFolder)folder).modifyACI(aSharedFolderUser,
                       OracleACI.READACI + OracleACI.WRITEACI);
                       System.out.println("Done Creating Shared Folder");
                     }
                }
}
```

単純なメッセージの追加

次の例では、特定のユーザーの受信ボックスに単純なメッセージを追加する方法を示します。

```
import java.io.*;
import java.net.InetAddress;
import java.util.Properties;
import java.util.Date;

import javax.mail.*;
import javax.mail.internet.*;

public class TestAppendMime
{

        static String password = null;
        static String user = null;
        static String host = null;
        static String ldapHost = null;
        static int ldapPort = -1;
        static String mbox = "INBOX";
```

```
public static void main(String[] argv) throws Exception{

    BufferedReader in =
                new BufferedReader(new InputStreamReader(System.in));

        for (int i = 0; i < argv.length; i++)
        {
                if (argv[i].equals("-U"))
                        user = argv[++i];
                else if (argv[i].equals("-P"))
                        password = argv[++i];
                else if (argv[i].equals("-D"))
                        ldapHost = argv[++i];
                else if (argv[i].equals("-I"))
                        ldapPort = Integer.parseInt(argv[++i]);
                else if (argv[i].equals("--")) {
                        i++;
                        break;
                }
                else if (argv[i].startsWith("-")) {
                        System.out.println(
                        "Usage: TestAppendMime [-D ldapHost] [-I ldapPort]
                        [-U user] [-P password]");
                        System.exit(1);
                }
                else {
                        break;
                }
        }
     Properties props = System.getProperties();

     // Set system properties - it is necessary to set up these
     // properties to obtain the database connect information
     // from oid. The database connect information is available only
     // to privileged users.
     // NOTE: The password may be different
     // for your installation.
     props.setProperty("oracle.mail.ldap.admin_dn","cn=umadmin,
     cn=emailservercontainer,cn=products,cn=oraclecontext");
     props.setProperty("oracle.mail.ldap.admin_password","umadmin");

     // Get a Session object
     Session session = Session.getDefaultInstance(props, null);
     session.setDebug(true);
```

```
        Store store = null;
        store = session.getStore("esmail");

        System.out.println(ldapHost + "\n" + ldapPort + "\n" + user + "\n" +
                password);
        store.connect(ldapHost, ldapPort, user, password);
        // Get record Folder.  Create if it does not exist.
        Folder folder = store.getFolder("INBOX");
        Message msg = new MimeMessage(session);
        msg.setFrom(new InternetAddress("oracle@oracle.com"));
        msg.setRecipients(Message.RecipientType.TO,
        InternetAddress.parse("testuser1@umdev.us.oracle.com", false));
        msg.setSubject("Welcome!!!");
        //collect(in, msg);
        msg.setText("Hello welcome\n");
        msg.setSentDate(new Date());

        System.out.println("Total Number of messages : " +
        folder.getMessageCount());

        Message[] msgs = new Message[1];
        msgs[0] = msg;
        folder.appendMessages(msgs);

        System.out.println("Total Number of messages : " +
        folder.getMessageCount());
        System.out.println("Mail was recorded successfully.");
    }

     public static void collect(BufferedReader in, Message msg)
                         throws MessagingException, IOException {
        String line;
        StringBuffer sb = new StringBuffer();
        while ((line = in.readLine()) != null) {
            sb.append(line);
            sb.append("\n");
        }

        // If the desired charset is known, you can use
        // setText(text, charset)
        msg.setText(sb.toString());
     }
}
```

基本的なフォルダ操作

次に、フォルダ操作の例を示します。この例では、フォルダの基本操作（フォルダの作成、フォルダへのメッセージのコピー、フォルダのリスト表示、フォルダ名の変更、メッセージの削除）を示します。

```
/
public class TestFolder
{
 static String password = null;
 static String user = null;
 static String host = null;
 static int port = -1;
 static String mbox = "INBOX";
 static String root = null;
 static boolean recursive = false;
 static String pattern = "*";
 static boolean verbose = false;
        static String namespace = null;

 public static void main (String argv[]) throws Exception
 {
  for (int i = 0; i < argv.length; i++)
  {
   if (argv[i].equals("-U"))
    user = argv[++i];
   else if (argv[i].equals("-P"))
    password = argv[++i];
   else if (argv[i].equals("-D"))
    host = argv[++i];
   else if (argv[i].equals("-I"))
    port = Integer.parseInt(argv[++i]);
   else if (argv[i].equals("--")) {
                 i++;
                 break;
              }
   else if (argv[i].startsWith("-")) {
                 System.out.println("Usage: TestFolder [-D ldap_host]
                 [-I ldap_port] [-U user] [-P password]");
                 System.exit(1);
   }
   else {
    break;
   }
  }

  Properties props = System.getProperties();
  Session session = Session.getDefaultInstance(props,null);
```

```
session.setDebug(false);

Store store = null;

store = session.getStore("esmail");

if (user != null && password != null && port != 0 && host != null){
store.connect(host, port, user, password);

              namespace = user.substring(0, user.indexOf('@'));

// Open the folder - after store has connected
Folder folder = store.getDefaultFolder();


if (folder == null) {
 System.out.println("Invalid folder");
 System.exit(1);
}
else {
                      /*** List folders ***************/
                      Folder[] f = folder.list(pattern);
                      for (int i = 0; i < f.length; i++)
                      {
                         System.out.println("Name: " + f[i].getName());
                         System.out.println("Full Name: " + f[i].getFullName());
                      }
                      /*** Create folder ***************/
                      System.out.println("Creating folder xyz");
                      Folder aFolder = store.getFolder("/" + namespace + "/xyz");

                      if (!aFolder.exists()) //create
                      {
                         System.out.println("Creating folder....");
                         aFolder.create(Folder.HOLDS_MESSAGES);
                      }
                      Folder aNewFolder = store.getFolder("/" + namespace +
                      "/temp");
                      /****** Renaming Folder ***************/
                      System.out.println("Renaming Folder to temp");
                      aFolder.renameTo(aNewFolder);
                      /***** List folder again **************/
                      f = folder.list(pattern);
                      for (int i = 0; i < f.length; i++)
                      {
                         System.out.println("Name: " + f[i].getName());
                         System.out.println("Full Name: " + f[i].getFullName());
```

```
}
/***** Copy messages into folder ****/
Folder mbox = store.getFolder("/" + namespace + "/INBOX");
mbox.open(Folder.READ_WRITE);
Message[] msgArray = new Message[3];
System.out.println("********" + mbox.getMessageCount());
//Message[] mboxArray = new Message[mbox.getMessageCount()];
for (int j = 0; j < 3; j++)
  msgArray[j] = mbox.getMessage(j+1);

//msgArray = mbox.getMessages(1,3);
mbox.copyMessages(msgArray, aFolder);
/***** Get message count **********/
System.out.println("Number of messages in folder  " +
    aFolder.getFullName() + " is " +
    aFolder.getMessageCount());

/***** delete messages in folder ****/
System.out.println("DELETING two MESSAGES ");
System.out.println("---> Number of messages in folder after
delete " + aFolder.getFullName() + " is " +
aFolder.getMessageCount());
aFolder.open(Folder.READ_WRITE);
Message aMessage = aFolder.getMessage(1);
Message aMessage2 = aFolder.getMessage(2);
aMessage.setFlag(Flags.Flag.DELETED,true);
aMessage2.setFlag(Flags.Flag.DELETED,true);
/******* expunge folder *************/
System.out.println("EXPUNGING");
aFolder.expunge();
aFolder.close(true);
System.out.println("---> Number of messages in folder after
expunge " + aFolder.getFullName() + " is " +
aFolder.getMessageCount());
/******* msg count of folder ********/


/***** List folder again **************/
System.out.println("***********LIST ***********");
f = folder.list(pattern);
for (int i = 0; i < f.length; i++)
{
   System.out.println("Name: " + f[i].getName());
   System.out.println("Full Name: " + f[i].getFullName());
}
/****** Delete Folder *******/
System.out.println("*********** DELETE ***********");
```

```
                    aFolder.delete(true);
                    /***** List folder again **************/
                    f = folder.list(pattern);
                    for (int i = 0; i < f.length; i++)
                    {
                       System.out.println("Name: " + f[i].getName());
                       System.out.println("Full Name: " + f[i].getFullName());
                    }

}
    }
       store.close();
          }

   public static void dumpFolder(Folder folder, boolean recurse, String tab)
   throws Exception
   {
       System.out.println(tab + "Name:      " + folder.getName());
       System.out.println(tab + "Full Name: " + folder.getFullName());
       System.out.println(tab + "URL:       " + folder.getURLName());

       if (verbose) {
            if (!folder.isSubscribed())
                System.out.println(tab + "Not Subscribed");

            if ((folder.getType() & Folder.HOLDS_MESSAGES) != 0) {
                if (folder.hasNewMessages())
                     System.out.println(tab + "Has New Messages");
                 System.out.println(tab + "Total Messages:  " +
                 folder.getMessageCount());
                 System.out.println(tab + "New Messages:    " +
                 folder.getNewMessageCount());
                 System.out.println(tab + "Unread Messages: " +
                 folder.getUnreadMessageCount());
             }
            if ((folder.getType() & Folder.HOLDS_FOLDERS) != 0)
                System.out.println(tab + "Is Directory");
       }

       System.out.println();

       if ((folder.getType() & Folder.HOLDS_FOLDERS) != 0) {
            if (recurse) {
                Folder[] f = folder.list();
                for (int i = 0; i < f.length; i++)
                     dumpFolder(f[i], recurse, tab + "    ");
            }
```

```
        }
    }
}
```

共有フォルダとメッセージのフェッチ

SharedFolderRead.java

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;
```

次の例では、基本的な共有フォルダとメッセージのフェッチ機能を示します。

```
public class SharedFolderRead
{
        static String user = null;
        static String password = null;
        static String ldapHost = null;
        static int ldapPort = -1;
        static String mbox = "INBOX";
        static String pattern = "*";

        public static void main (String argv[]) throws Exception
        {
                for (int i = 0; i < argv.length; i++)
                {
                        if (argv[i].equals("-U"))
                                user = argv[++i];
                        else if (argv[i].equals("-P"))
                                password = argv[++i];
                        else if (argv[i].equals("-D"))
                                ldapHost = argv[++i];
                        else if (argv[i].equals("-I"))
                                ldapPort = Integer.parseInt(argv[++i]);
                        else if (argv[i].equals("--")) {
                                i++;
                                break;
                        }
                        else if (argv[i].startsWith("-")) {
                                System.out.println(
                                "Usage: SharedFolderRead [-D ldap_host]
                                [-I ldap_port] [-U user] [-P password]");
                                System.exit(1);
                        }
```

```
        else {
                break;
        }
}

Properties props = System.getProperties();

// Set system properties - it is necessary to set up these
// properties to obtain the database connect information
// from oid. The database connect information is available only
// to privileged users.
// NOTE: The passwordmay be different
// for your installation.
props.setProperty("oracle.mail.ldap.admin_dn","cn=umadmin,
cn=emailservercontainer,cn=products,cn=oraclecontext");
props.setProperty("oracle.mail.ldap.admin_password","umadmin");

Session session = Session.getDefaultInstance(props,null);
session.setDebug(false);

Store store = null;

store = session.getStore("esmail");

if (user != null && password != null && ldapPort != 0 && ldapHost
   != null){
        store.connect(ldapHost, ldapPort, user, password);

        Folder aLocalFolder = store.getFolder(mbox);
if (aLocalFolder == null) {
        System.out.println("NULL FOLDER " + mbox);
        System.exit(1);
}
else {
        System.out.println("Folder Name: " +
        aLocalFolder.getFullName());
}

try {
        aLocalFolder.open(Folder.READ_WRITE);
} catch (MessagingException ex) {
        aLocalFolder.open(Folder.READ_ONLY);
}

int totalMessages = aLocalFolder.getMessageCount();
```

```
System.out.println("Total Number of messages in folder " +
        aLocalFolder.getFullName() + " is " + totalMessages);

Folder[] aSharedNSArray = store.getSharedNamespaces();

if (aSharedNSArray == null) {
        System.out.println("No shared namespaces");
        System.exit(1);
}
else {
        //For each namespace, list
        System.out.println("Number of shared namespace : " +
        aSharedNSArray.length);

        Folder folder = null;
        for (int i = 0; i < aSharedNSArray.length; i++)
        {
           folder = aSharedNSArray[i];
           System.out.println("Folder fullname : " +
           folder.getFullName());
        }
                Folder[] f = folder.list(pattern);
                int aNum =0;
                for (int j = 0; j < f.length; j++)
                {
                        System.out.println("Name:      " +
                        f[j].getName());
                        System.out.println("Full Name: " +
                        f[j].getFullName());
                        aNum = j;

                try {
                        f[aNum].open(Folder.READ_WRITE);
                } catch (MessagingException ex) {
                        f[aNum].open(Folder.READ_ONLY);
                }

         System.out.println("Shared Folder fullname : " +
         f[aNum].getFullName());

         // Select and fetch messages from Shared Folder
         int total_messages = f[aNum].getMessageCount();
         System.out.println("Number of messages in " +
         f[aNum].getFullName() + " is " +
         f[aNum].getMessageCount());

         if (total_messages == 0) {
```

```
                        System.out.println("Empty folder " +
                        f[aNum].getFullName());
                        f[aNum].close(false);
                    }
                    else {
                           int my_uid = 0;
                           Message[] msgs = f[aNum].getMessages();
                           if (msgs != null)
                                   System.out.println("Number of messages : " +
                                   msgs.length);
                           for (int i = 0; i < msgs.length; i++)
                           {
                             my_uid = msgs[i].getMessageNumber();
                             System.out.println("MSG_NUMBER : " + my_uid);
                             if (my_uid > 0)
                             {
                               System.out.println("Retrieving msg_num : " +
                               my_uid);
                               Message msg = f[aNum].getMessage(my_uid);
                               dumpPart(msg);
                              }
                           }

                          // Copy message from shared folder to local folder
                          //System.out.println("----- BEFORE COPY -------");
                          //f[aNum].copyMessages(msgs, aLocalFolder);
                          //System.out.println("----- AFTER COPY -------");
                    }
                }
              }
            }
            store.close();
}

//Fetch message part by part
public static void dumpPart(Part p) throws Exception
{
    if (p instanceof Message)
      //pr("Envelope out of order------------>");
      dumpEnvelope((Message)p);

    //InputStream is = p.getInputStream();
    //System.out.println("SIZE of INPUT STREAM : " + is.available());
    System.out.println("----------------------------");
    pr("CONTENT-TYPE: " + p.getContentType());

    if (p.isMimeType("text/plain")) {
```

```
            pr("This is plain text");
            pr("--------------------------");
            System.out.println((String)p.getContent());
        } else if (p.isMimeType("multipart/*")) {
            pr("This is a Multipart");
            pr("--------------------------");
            Multipart mp = (Multipart)p.getContent();
            int count = mp.getCount();
            for (int i = 0; i < count; i++)
               dumpPart(mp.getBodyPart(i));
            } else if (p.isMimeType("message/rfc822")) {
               pr("This is a Nested Message");
               pr("--------------------------");
               dumpPart((Part)p.getContent());
            }
            else {
              Object o = p.getContent();
              if (o instanceof String) {
               pr("This is a string");
               pr("--------------------------");
               System.out.println((String)o);
              }
             else if (o instanceof InputStream) {
                  pr("This is just an input stream");
                  pr("--------------------------");
                  InputStream is = (InputStream)o;
                  //is = (InputStream)o;
                  int c;
                  while ((c = is.read()) != -1)
                     System.out.write(c);
               } else {
                  pr("This is an unknown type");
                  pr("--------------------------");
                  pr(o.toString());
               }
             }
        }

    public static void dumpEnvelope(Message m) throws Exception {
        pr("This is the message envelope");
        pr("--------------------------");
        Address[] a;

        // FROM
        if ((a = m.getFrom()) != null) {
            for (int j = 0; j < a.length; j++)
                pr("FROM: " + a[j].toString());
```

```
            }

            // TO
            if ((a = m.getRecipients(Message.RecipientType.TO)) != null) {
                    for (int j = 0; j < a.length; j++)
                            pr("TO: " + a[j].toString());
            }

            // SUBJECT
                    pr("SUBJECT: " + m.getSubject());

            // DATE
            Date d = m.getSentDate();
            pr("SendDate: " +
                            (d != null ? d.toString() : "UNKNOWN"));
        }

        public static void pr(String s){
            System.out.println(s);
        }
        ////--------------
    }
```

ディレクトリ管理 API

ディレクトリ管理 API は、メール・ユーザー、パブリック配布リスト、プライベート・アドレス帳エントリ（連絡先情報とプライベート配布リスト）などの様々なエントリの作成、アクセスおよび管理に使用できる Java クラスのセットです。エントリは、Oracle Internet Directory の特定のドメインに格納されます。

この項では、次の項目について説明します。

- ディレクトリ・コンポーネント
- 認証
- メタデータの取得と検証
- ディレクトリ管理のコード例

関連資料： ディレクトリ管理 API の詳細は、「Oracle Email API Reference」（Java Doc）を参照してください。

ディレクトリ・コンポーネント

Oracle Email では、電子メール・システムに複数のドメインを含めることができます。各ドメインには、メール・ユーザーとパブリック配布リストが存在します。特定のドメインのメール・ユーザーはそのドメインの有効ユーザーで、電子メールを送受信し、公開されたすべての電子メール・サーバー機能を使用できます。

パブリック配布リストは、独自の電子メール ID を持つ、電子メール ID または他のメーリング・リストのグループを含むメーリング・リストです。電子メールをパブリック配布リストに送ると、リストのすべてのメンバーが電子メールを受信します。有効なメール・ユーザーは、任意のパブリック配布リストに登録できます。

プライベート・アドレス帳エントリは、特定のメール・ユーザーに属するプライベート連絡先とプライベート・メーリング・リストで構成されます。

プライベート配布リストは、連絡先の電話番号、電子メール ID、住所などのプライベート連絡先情報で構成されます。ユーザーは、Webmail からプライベート・アドレス帳エントリを使用して、電子メールを送受信できます。これらのエントリは、カレンダ・アプリケーションでも使用されます。

認証

コール元がディレクトリ・コンポーネントにアクセスするには、その前に `oracle.mail.OESContext` クラスを使用して Oracle Internet Directory の認証を受ける必要があります。認証後、トラステッド・セッションを表す `oracle.mail.OESContext` インスタンスをすべてのディレクトリ API に渡す必要があります。

認証の例

次の例では、デバッグ・オプションをオフにしたアプリケーションの認証方法を示します。

```
OESContext oesctx = new OESContext(DirectoryConstants.DS_CALLERTYPE_APP, false);
//Authenticate to the directory
oesctx.authenticate(null, args[0]);
```

メタデータの取得と検証

ディレクトリでエントリを作成する前に、コール元はディレクトリからそのエントリのメタデータを取得する必要があります。特定のエントリのメタデータは、コール元がエントリを作成するために設定する必要がある必須属性とオプションの属性で構成されます。メタデータには、構文、属性の多重度、属性のデフォルト値（ディレクトリでデフォルト値が設定されている場合）などのすべての属性に関する情報も含まれます。

コール元がメタデータ・オブジェクトで属性値を設定すると、コール元がそのエントリに存在する属性の値を設定したかどうかを確認する検証が実行されます。メタデータを使用する UI ベースのアプリケーションでは、コール元は入力したデータに対して入力の検証を実行できます。

例

次の例では、メタデータの取得方法を示します。ldapobj が DirectoryObject クラスのインスタンスであることを前提にしています。

```
//Getting the mandatory attributes from the metadata
if (ldapobj.getMandatoryAttribs() != null) {
  Enumeration enum = ldapobj.getMandatoryAttribs().elements();
  while (enum.hasMoreElements()) {
    String attr = enum.nextElement().toString(); // Name of the attribute

    //Retrieve the metadata for this attribute
    DirectoryAttributeMetaData mdata = ldapobj.getMetaData(attr);
    // The multiplicity of the attribute returns "SINGLE" or "MULTIPLE"
    String mult = mdata.getMultiplicity();
    // Returns the syntax of the string, "String", "byte", "boolean" or "int"
    String syntax = mdata.getAttributeType();
    //Returns a vector of String values if any default has been set, else returns
    //null
    Vector defaultvals = mdata.getDefaultValues();
  }
}
```

同様に、ldapobj.getOptionalAttribs() メソッドはオプションの属性リストを返し、オプションの属性とメタデータを取得できます。メタデータの取得後、ユーザーは次の方法で属性の値を設定できます。エントリを作成する際、DirectoryObject クラスの setAttributeValue メソッドを使用して属性値を設定できます。

例

この例では、mdata.getDefaultValues() が NULL でない場合、telephonenumber 属性の値をデフォルト値に設定します。

```
ldapobj.setAttributeValue("telephonenumber", mdata.getDefaultValues());

Vector newVals = new Vector();
newVals.add("408 7394050");
newVals.add("650 7394050");
ldapobj.setAttributeValue("telephonenumber", newVals);
```

エントリを変更する際、DirectoryObject クラスの modifyAttributeValue メソッドを使用して属性値を設定できます。コール元は変更のタイプを指定する必要があります。

指定できる変更は次のとおりです。

- DirectoryConstants.DS_MODIFY_ADD: 指定された値セットを既存の値に追加します。
- DirectoryConstants.DS_MODIFY_DELETE: 指定された値セットを既存の値から削除します。

- DirectoryConstants.DS_MODIFY_REPLACE: 既存のすべての値を新しい値セットに置き換えます。

例

この例では、telephonenumber 属性に新しい 2 つの値を追加します。

```
Vector newVals = new Vector();
newVals.add("408 7394050");
newVals.add("650 7394050");
ldapobj.modifyAttributeValue("telephonenumber", newVals,
DirectoryConstants.DS_MODIFY_ADD);
```

ディレクトリ管理のコード例

この項では、ディレクトリ管理 API について次の例を示します。

注意： これらの例を実行するには、CLASSPATH 環境変数に jndi.jar、ldap.jar、providerutil.jar、classes12.zip、repository.jar、esldap.jar および escommon.jar が含まれている必要があります。

メール・ユーザーの例

次の例では、メール・ユーザー API を使用して、メール・ユーザーを作成、変更、参照および削除する方法を示します。

メール・ユーザーの作成例：

この例では、メール・ユーザーを作成します。メール・ユーザーの属性はデフォルト値に設定されます。この例では、orclMailVoiceQuota パラメータを指定して、属性の値をデフォルト値以外の値に設定する方法を示します。

```
import java.util.Enumeration;
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryAttributeMetaData;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;

class CreateMailUser
{
```

```
public static void main(String[] args)
{
    if (args.length < 3)
    {
        System.out.println("Usage: java CreateMailUser <ORAHOME> <mailid>
        <publicuserdn>");
        System.exit(1);
    }

    try
    {
        OESContext oesctx = new OESContext(
                                    DirectoryConstants.DS_CALLERTYPE_APP,
                                    false);

        //Authenticate to the directory
        if (oesctx.authenticate(null, args[0]) != true)
        {
            System.err.println("APP authentication failed");
            System.exit(1);
        }
        else
        {
            DirectoryAccess access = new DirectoryAccess();
            String mail = args[1];
            String domain = null;

            if (mail.indexOf('@') != -1)
            {
                domain = mail.substring(mail.indexOf('@') + 1,
                                    mail.length());

                //Retrieve Metadata
                DirectoryObject ldapobj = access.GetMailUserMetaData(
                                                oesctx, domain);
                System.out.println("Retrieved Metadata.");
                System.out.println("Setting mandatory attributes of the
                mailuser.");

                //Set mandatory and optional attributes of a mailuser to
                //defaults
                if (ldapobj.getMandatoryAttribs() != null)
                {
                    Enumeration enum = ldapobj.getMandatoryAttribs().elements();

                    while (enum.hasMoreElements())
                    {
```

```
        String attr = enum.nextElement().toString();
        DirectoryAttributeMetaData mdata =
                ldapobj.getMetaData(attr);

        if (attr.equalsIgnoreCase("mail"))
        {
            Vector vals = new Vector();
            vals.add(mail);
            ldapobj.setAttributeValue(attr, vals);
            continue;
        }
        else
        {
            ldapobj.setAttributeValue(attr,
            mdata.getDefaultValues());
            continue;
        }
    }
}

System.out.println("Setting optional attributes of the
mailuser.");

if (ldapobj.getOptionalAttribs() != null)
{
    Enumeration enum = ldapobj.getOptionalAttribs().elements();

    while (enum.hasMoreElements())
    {
        String attr = enum.nextElement().toString();
        DirectoryAttributeMetaData mdata =
                ldapobj.getMetaData(attr);

        if (mdata.getDefaultValues() != null)
            ldapobj.setAttributeValue(attr,
            mdata.getDefaultValues());
    }
}

//Setting voice quota to 40MB
Vector vals = new Vector();
vals.add("40000000");
ldapobj.setAttributeValue("orclMailVoiceQuota", vals);
System.out.println("Creating Mailuser .." + mail);

//Create the mailuser
access.CreateMailUser(oesctx, args[2], ldapobj);
```

```
                    System.out.println("Created Mailuser ..");
                }
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

メール・ユーザーの変更例 :

この例では、メール・ユーザーを参照します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryAttributeMetaData;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;



class ModifyMailUser
{
    public static void main(String[] args)
    {
        if (args.length < 2)
        {
            System.out.println("Usage: java ModifyMailUser <ORAHOME> <mailid>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                    DirectoryConstants.DS_CALLERTYPE_APP,
                                    false);
```

```
                //Authenticate to the directory
                if (oesctx.authenticate(null, args[0]) != true)
                {
                    System.err.println("APP authentication failed");
                    System.exit(1);
                }
                else
                {
                    DirectoryAccess access = new DirectoryAccess();
                    DirectoryObject ldapObj = access.LookupMailUser(oesctx,
                                                                    args[1]);

                    if (ldapObj != null)
                    {
                        //Retrieving Metadata of orclMailQuota attribute
                        DirectoryAttributeMetaData mdata = ldapObj.getMetaData(
                                                                "orclMailQuota");

                        //Retrieving the values of this attribute
                        Vector values = mdata.getDefaultValues();

                        if (values != null)
                        {
                            for (int i = 0; i < values.size(); i++)
                                System.out.println("orclMailQuota Values --> " +
                                                   (String)values.elementAt(i));
                        }

            //Use DirectoryConstants.DS_MODIFY_DELETE to delete the given set of
            //values from the existing  values of the attribute
            //DirectoryConstants.DS_MODIFY_REPLACE to replace the existing values
            // of the attribute with the new set of values
            //DirectoryConstants.DS_MODIFY_ADD to add the given set of values to
            //the existing values  of the attribute
            //Setting mail quota to 100MB, this replaces the existing mail quota
            //with the new values
                        values = new Vector();
                        values.add("100000000");
                        ldapObj.modifyAttributeValue("orclMailQuota", values,
                                                     DirectoryConstants.DS_MODIFY_
                                                     REPLACE);
                        System.out.println("Modifying Mailuser ..");
                        access.ModifyMailUser(oesctx, args[1], ldapObj);
                        System.out.println("Modified Mailuser ..");
                    }
                }
```

```
            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

メール・ユーザーの削除例 :

この例では、メール・ユーザーを削除します。

```
import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class DeleteMailUser
{
    public static void main(String[] args)
    {
        if (args.length < 2)
        {
            System.out.println("Usage: java DeleteMailUser <ORAHOME> <mailid>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                    DirectoryConstants.DS_CALLERTYPE_APP,
                                    false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Deleting User " + args[1]);
```

```
                //Deleting the mailuser
                access.DeleteMailUser(oesctx, args[1]);
                System.out.println("Deleted User ");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

メール・ユーザーの問合せ例 :

この例では、ドメイン内のすべてのメール・ユーザーに問合せを実行します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class QueryMailUsers
{
    public static void main(String[] args)
    {
        if (args.length < 2)
        {
            System.out.println("Usage: java QueryMailUsers <ORAHOME> <domain>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
```

```
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Querying Users ");

                //Query for all mailusers, similarly other search criteria could
                //also
                //be given for the query, for example, t*
                Vector users = access.QueryMailUsers(oesctx, "*", args[1]);

                if (users != null)
                {
                    for (int i = 0; i < users.size(); i++)
                        System.out.println("User " + (i + 1) + " " +
                                           (String)users.elementAt(i));
                }
                else
                {
                    System.out.println("Nothing matches the search criteria ");
                }
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

配布リストの作成例 :

この例では、ディレクトリにパブリック配布リストを作成します。配布リストの属性はデフォルト値に設定されます。

```
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;

import oracle.mail.OESContext;
```

```
import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryAttributeMetaData;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;



class CreateDistributionList
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java CreateDistributionList <ORAHOME>
            <domain> <dl_mailid> <user_member_mailid>");
            System.exit(1);
        }


        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                DirectoryAccess access = new DirectoryAccess();
                DirectoryObject ldapobj = access.GetDistributionListMetaData(
                                                    oesctx, args[1]);
                System.out.println("Retrieved Metadata.");
                System.out.println("Setting mandatory attributes of the DL.");

                if (ldapobj.getMandatoryAttribs() != null)
                {
                    Enumeration enum = ldapobj.getMandatoryAttribs().elements();

                    while (enum.hasMoreElements())
                    {
                        String attr = enum.nextElement().toString();
```

```
        DirectoryAttributeMetaData mdata = ldapobj.getMetaData(
                                                    attr);

        if (attr.equalsIgnoreCase("mail"))
        {
            Vector vals = new Vector();
            vals.add(args[2]);
            ldapobj.setAttributeValue(attr, vals);
            continue;
        }
        else
        {
            ldapobj.setAttributeValue(attr,
                                    mdata.getDefaultValues());
            continue;
        }
    }
}

System.out.println("Setting optional attributes of the DL.");

if (ldapobj.getOptionalAttribs() != null)
{
    Enumeration enum = ldapobj.getOptionalAttribs().elements();

    while (enum.hasMoreElements())
    {
        String attr = enum.nextElement().toString();
        DirectoryAttributeMetaData mdata = ldapobj.getMetaData(
                                                    attr);

        if (mdata.getDefaultValues() != null)
            ldapobj.setAttributeValue(attr,
            mdata.getDefaultValues());
    }
}

System.out.println("Creating DL .." + args[2]);

//Creating DL with one user member, similarly other types of members
//could also be added
Hashtable members = new Hashtable();
Vector usrs = new Vector();
usrs.add(args[3]);
members.put(DirectoryConstants.DS_USER, usrs);
```

```
                // returns a hashtable of elements that couldn't be added
                Hashtable failed = access.CreateDistributionList(oesctx,
                                                                 ldapobj,
                                                                 members);
                System.out.println("Created DL ..");

                if (failed != null)
                {
                    System.out.println("Some member additions failed.");
                }
                else
                {
                    System.out.println("Successfully added all members.");
                }
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

配布リストの変更例 :

この例では、特定の配布リストを参照し、この既存のパブリック配布リストの属性を変更します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;


class ModifyDistributionList
{
    public static void main(String[] args)
    {
        if (args.length < 2)
        {
```

```
    System.out.println("Usage: java ModifyDistributionList <ORAHOME>
    <dl_mailid>");
    System.exit(1);
}

try
{
    OESContext oesctx = new OESContext(
                                DirectoryConstants.DS_CALLERTYPE_APP,
                                false);

    //Authenticate to the directory
    if (oesctx.authenticate(null, args[0]) != true)
    {
        System.err.println("APP authentication failed");
        System.exit(1);
    }
    else
    {
        DirectoryAccess access = new DirectoryAccess();
        System.out.println("Looking up DL.. " + args[1]);

        DirectoryObject ldapObj = access.LookupDistributionList(oesctx,
        args[1]);
        System.out.println("Found DL \n Printing Old Values of
        orclMailGroupInfoText");

        Vector values = ldapObj.getAttributeValue("orclmailgroupinfotext");

        if (values != null)
        {
            for (int i = 0; i < values.size(); i++)
                System.out.println("orclMailGroupInfoText Values --> " +
                                    (String)values.elementAt(i));
        }
        else
        {
            System.out.println("No value set for orclMailGroupInfoText");
        }

        System.out.println("Adding orclmailgroupinfotext to this DL");

        //Use DirectoryConstants.DS_MODIFY_DELETE to delete the given set of
        //values from the existing  values of the attribute
        //DirectoryConstants.DS_MODIFY_REPLACE to replace the existing
        //values of the attribute with the new set of values
        //DirectoryConstants.DS_MODIFY_ADD to add the given set of values to
```

```
                //the existing values  of the attribute
                //Setting orclmailgroupinfotext to some new values
                values = new Vector();
                values.add("Test DL");
                ldapObj.modifyAttributeValue("orclmailgroupinfotext", values,
                                             DirectoryConstants.DS_MODIFY_ADD);
                System.out.println("Modifying DL ..");
                access.ModifyDistributionList(oesctx, args[1], ldapObj);
                System.out.println("Modified DL ..");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

配布リストの削除例 :

この例では、パブリック配布リストを削除します。

```
import java.util.Enumeration;
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;


class DeleteDistributionList
{
    public static void main(String[] args)
    {
        if (args.length < 2)
        {
            System.out.println("Usage: java DeleteDistributionList <ORAHOME>
            <mailid>");
            System.exit(1);
        }

        try
        {
```

```
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Deleting DL " + args[1]);

                //Deleting the DL
                access.DeleteDistributionList(oesctx, args[1]);
                System.out.println("Deleted DL");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

配布リストの解決例 :

この例では、パブリック配布リストを解決する方法を示します。

```
import java.util.Hashtable;
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;


class ResolveDistributionList
{
    public static void main(String[] args)
    {
```

```
if (args.length < 2)
{
    System.out.println("Usage: java ResolveDistributionList <ORAHOME>
    <dl_mailid>");
    System.exit(1);
}

try
{
    OESContext oesctx = new OESContext(
                                DirectoryConstants.DS_CALLERTYPE_APP,
                                false);

    //Authenticate to the directory
    if (oesctx.authenticate(null, args[0]) != true)
    {
        System.err.println("APP authentication failed");
        System.exit(1);
    }
    else
    {
        DirectoryAccess access = new DirectoryAccess();
        Hashtable members = access.ResolveDistributionList(oesctx,
                                                            args[1]);
        System.out.println("Printing Members..");

        //Key of the hashtable is the type of member and value is a vector
        //of values
        if (members != null)
        {
            Vector users = (Vector)members.get(DirectoryConstants.DS_USER);

            if (users != null)
            {
                for (int i = 0; i < users.size(); i++)
                    System.out.println("User Member :: " +
                                        users.elementAt(i).toString());
            }

            Vector dls = (Vector)members.get(DirectoryConstants.DS_LIST);

            if (dls != null)
            {
                for (int i = 0; i < dls.size(); i++)
                    System.out.println("DL Member :: " +
                                        dls.elementAt(i).toString());
            }
```

```
                    Vector foreign = (Vector)members.get(DirectoryConstants.
                    DS_FOREIGN);

                    if (foreign != null)
                    {
                        for (int i = 0; i < foreign.size(); i++)
                            System.out.println("DL Member :: " +
                                               foreign.elementAt(i).toString());
                    }

                    Vector aliases = (Vector)members.get(DirectoryConstants.
                    DS_ALIAS);

                    if (aliases != null)
                    {
                        for (int i = 0; i < aliases.size(); i++)
                            System.out.println("DL Member :: " +
                                               aliases.elementAt(i).toString());
                    }
                }
                else
                {
                    System.out.println("DL doesn't contain any member");
                }
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

プライベート・アドレス帳連絡先情報の例

次の例では、プライベート・アドレス帳連絡先情報 API を使用して、連絡先を作成、変更、参照、解決および検索する方法を示します。

```
Getting Contact Info Metadata And Creating A New Contact Info
**/

ESDSLdapObject ldapobj = dirAccess.GetContactInfoMetaData(oesctx);
```

この例では、新規連絡先の名前と電子メール ID を設定します。他の属性も同じ方法で設定できます。

```
ldapobj.setAttributeValue("name","myfriend1");

ldapobj.setAttributeValue("orclmailemail","test@hotmail.com");

// Now Create The Contact

dirAccess. CreateContactInfo(oesctx, ldapobj);

/**

Looking up a Contact Info

**/

ldapobj = dirAccess.LookupContactInfo(oesctx,"myfriend1");

/***

Modifying a Contact

**/

//Modifying the given name of the contact

Vector modify = new Vector();

modify.add ("myfriend_name");

ldapobj.modifyAttributeValue("givenname", modify, ESDSConstants.DS_MODIFY_ADD);

dirAccess.ModifyContactInfo(oesctx, "myfriend1",ldapobj);

/***

Delete A Contact Info

***/

dirAccess.DeleteContactInfo(oesctx,"friend ");

/**

Get All Contacts
```

```
**/

String[] contacts = dirAccess.GetAllContacts(oesctx);

/**

Get All Contacts For a Given Filter

**/

contacts = dirAccess.SearchContacts(oesctx,"name=t*");
```

プライベート配布リストの例

次の例では、プライベート配布リスト API を使用して、連絡先を作成、変更、参照、解決および検索する方法を示します。

連絡先の作成例 :

この例では、特定のユーザーにプライベート連絡先を作成します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;


class CreateContact
{
    public static void main(String[] args)
    {
        if (args.length < 3)
        {
            System.out.println("Usage: java CreateContact <ORAHOME> <user_mailid>
            <user_password>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);
```

```
            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                DirectoryObject ldapobj = access.GetContactInfoMetaData(
                                                clientCtx);
                System.out.println("Retrieved Metadata.");

                //Setting some attributes of the contact
                ldapobj.setAttributeValue("name", "friend1");
                ldapobj.setAttributeValue("givenname", "myfriend");
                ldapobj.setAttributeValue("orclmailemail", "test1@abc.com");

                Vector tel = new Vector();
                tel.add("405 7777777");
                tel.add("650 7777777");

                ldapobj.setAttributeValue("telephonenumber", tel);
                access.CreateContactInfo(clientCtx, ldapobj);
                System.out.println("Created Contact ..");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

連絡先の変更例：

この例では、連絡先を参照し、変更します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;


class ModifyContact
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java ModifyContact <ORAHOME> <user_mailid>
            <user_password> <user_contact_name>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                DirectoryObject ldapobj = access.LookupContactInfo(clientCtx,
                                                                    args[3]);
                System.out.println("Found Contact");
```

```
                Vector values = ldapobj.getAttributeValue("telephonenumber");
                System.out.println("Printing old values of telephonenumber of the
                contact");

                if (values != null)
                {
                    for (int i = 0; i < values.size(); i++)
                        System.out.println("telephonenumber Values --> " +
                                           (String)values.elementAt(i));
                }
                else
                {
                    System.out.println("No values set for telephonenumber");
                }

                values = new Vector();
                values.add("6506070000");
                ldapobj.modifyAttributeValue("telephonenumber", values,
                                             DirectoryConstants.DS_MODIFY_ADD);

                Vector tel = new Vector();
                tel.add("408 7431234");
                tel.add("650 0000000");
                ldapobj.setAttributeValue("mobile", tel);
                access.ModifyContactInfo(clientCtx, ldapobj);
                System.out.println("Modified Contact ..");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

連絡先の削除例：

この例では、連絡先情報を削除します。

```
import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class DeleteContact
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java DeleteContact <ORAHOME> <user_mailid>
            <user_password> <user_contact_name>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Deleting Contact " + args[3]);

                //Deleting the contact
                access.DeleteContactInfo(clientCtx, args[3]);
                System.out.println("Deleted Contact");
```

```
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

連絡先の問合せ例 :

この例では、ユーザーのプライベート・アドレス帳の連絡先の様々な問合せ方法を示します。

```
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class QueryContacts
{
    public static void main(String[] args)
    {
        if (args.length < 3)
        {
            System.out.println("Usage: java QueryContacts <ORAHOME> <user_mailid>
            <user_password>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
```

```
                {
                    System.err.println("APP authentication failed");
                    System.exit(1);
                }
                else
                {
                    //Authenticate the user now
                    OESContext clientCtx = new OESContext(DirectoryConstants.
                    DS_CALLERTYPE_MAILUSER);
                    clientCtx.authenticate(args[1], args[2], oesctx);

                    DirectoryAccess access = new DirectoryAccess();
                    System.out.println("Retrieving All Contacts");

                    //Retrieving all contacts in the user's addressbook
                    String[] contacts = access.GetAllContacts(clientCtx);

                    if (contacts != null)

                        for (int i = 0; i < contacts.length; i++)
                            System.out.println("Contact " + (i + 1) + " " +
                                               contacts[i]);

                    //Retrieves all Contacts for the user logged on for the given search
                    //criteria. "name=t*" returns all the contacts starting with "t".
                    System.out.println("Retrieving Contacts Matching name=friend*");
                    contacts = access.SearchContacts(clientCtx, "name=friend*");

                    if (contacts != null)

                        for (int i = 0; i < contacts.length; i++)
                            System.out.println("Contact " + (i + 1) + " " +
                                               contacts[i]);
                }

                System.gc();
            }
            catch (Exception e)
            {
                System.err.println("Error Occured " + e.getMessage());
                System.gc();
            }
        }
    }
```

プライベート・リストの例

次の例では、プライベート・リスト API を使用して、リストを作成、変更、参照および削除する方法を示します。

プライベート・リストの作成例 :

この例では、特定のユーザーにプライベート配布リストを作成します。

```
import java.util.Vector;
import oracle.mail.OESContext;
import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;

class CreatePrivateDL
{
    public static void main(String[] args)
    {
        if (args.length < 3)
        {
            System.out.println("Usage: java CreatePrivateDL <ORAHOME> <user_mailid>
            <user_password>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);
```

```
                DirectoryAccess access = new DirectoryAccess();
                DirectoryObject ldapobj = access.GetPrivateListMetaData(
                                                clientCtx);
                System.out.println("Retrieved Metadata.");

                //Setting some attributes of the dl
                ldapobj.setAttributeValue("name", "friends");
                ldapobj.setAttributeValue("orclmailemail", "School Friends");

                Vector vect = new Vector();
                vect.add("test1@abc.com");
                vect.add("test2@abc.com");
                vect.add("test1@hotmail.com");
                ldapobj.setAttributeValue("orclmailemail", vect);
                access.CreatePrivateList(clientCtx, ldapobj);
                System.out.println("Created 1st List ..");

                //Now it will create another list and add the first list to this
                //one.
                ldapobj = access.GetPrivateListMetaData(clientCtx);

                //Setting some attributes of the dl
                ldapobj.setAttributeValue("name", "staff");
                ldapobj.setAttributeValue("orclmailemail", "Staff");
                vect = new Vector();
                vect.add("test1@yahoo.com");
                vect.add("test2@yahoo.com");
                ldapobj.setAttributeValue("orclmailemail", vect);

                //Add friends to staff, Similarly contact infos also can be added
                //to a private DL
                ldapobj.setAttributeValue("uniquemember", "friends");
                access.CreatePrivateList(clientCtx, ldapobj);
                System.out.println("Created 2nd List ..");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

プライベート・リストの変更例 :

この例では、プライベート配布リストを参照し、変更します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;
import oracle.mail.sdk.ldap.DirectoryObject;



class ModifyPrivateDL
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java ModifyPrivateDL <ORAHOME> <user_mailid>
            <user_password> <PrivateDL_name>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                DirectoryObject ldapobj = access.LookupPrivateList(clientCtx,
                                                                    args[3]);
```

```
                System.out.println("Found PrivateDL");

                Vector values = ldapobj.getAttributeValue("orclmailemail");
                System.out.println("Printing old values of mail ids of the PrivateDL
                members");

                if (values != null)
                {
                    for (int i = 0; i < values.size(); i++)
                        System.out.println("Mail ids of members --> " +
                                           (String)values.elementAt(i));
                }

                values = new Vector();
                values.add("user1@test.net");
                values.add("user2@oracle.com");
                ldapobj.modifyAttributeValue("orclmailemail", values,
                                             DirectoryConstants.DS_MODIFY_ADD);
                access.ModifyPrivateList(clientCtx, ldapobj);
                System.out.println("Modified PrivateDL ..");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

プライベート・リストの削除例 :

この例では、ユーザーのアドレス帳からプライベート配布リストを削除します。

```
import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;


class DeletePrivateDL
{
    public static void main(String[] args)
    {
        if (args.length < 4)
```

```
        {
            System.out.println("Usage: java DeletePrivateDL <ORAHOME> <user_mailid>
            <user_password> <user_DL_name>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Deleting private list " + args[3]);

                //Deleting the DL
                access.DeletePrivateList(clientCtx, args[3]);
                System.out.println("Deleted private list");
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

プライベート・リストの問合せ例 :

この例では、問合せ条件に基いてプライベート配布リストに問合せを実行します。

```
import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;


class QueryPrivateDLs
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java QueryPrivateDLs <ORAHOME> <user_mailid>
            <user_password> <ldap_query_filter_(eg. name=friend*)>");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Retrieving All DLs");

                //Retrieve all private dls present in the user's addressbook
                String[] dls = access.GetAllPrivateLists(clientCtx);
```

```
                if (dls != null)

                    for (int i = 0; i < dls.length; i++)
                        System.out.println("DL " + (i + 1) + " " + dls[i]);

                //Retrieves all DLs for the user logged on for the given search
                //criteria. "name=t*" returns all the DLs starting with "t".
                System.out.println("Retrieving DLs Matching " + args[3]);
                dls = access.SearchPrivateLists(clientCtx, args[3]);

                if (dls != null)

                    for (int i = 0; i < dls.length; i++)
                        System.out.println("DL " + (i + 1) + " " + dls[i]);
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

プライベート・リストの解決例 :

この例では、特定のプライベート・リストで電子メール・アドレスを解決します。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class ResolvePrivateList
{
    public static void main(String[] args)
    {
        if (args.length < 4)
        {
            System.out.println("Usage: java ResolvePrivateLists <ORAHOME>
            <user_mailid> <user_password> <private_dl_name>");
```

```
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                    DirectoryConstants.DS_CALLERTYPE_APP,
                                    false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);

                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Resolving this DL for email ids..");

                Vector resolved = access.ResolvePrivateList(clientCtx, args[3],
                                                        "orclmailemail");

                if (resolved != null)

                    for (int i = 0; i < resolved.size(); i++)
                        System.out.println("Email" + (i + 1) + " " +
                                        (String)resolved.elementAt(i));
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
}
```

ユーザー状態の取得例 :

この例では、ユーザー・コンテキストからユーザー状態を取得します。複数の電子メール・サーバーが稼働しているときに、コール元アプリケーションがどのシステムに接続するかを判断する際に便利です。

```
import java.util.Vector;

import oracle.mail.OESContext;

import oracle.mail.sdk.ldap.DirectoryAccess;
import oracle.mail.sdk.ldap.DirectoryConstants;
import oracle.mail.sdk.ldap.DirectoryException;



class RetrieveUserState
{
    public static void main(String[] args)
    {
        if (args.length < 3)
        {
            System.out.println("Usage: java RetrieveUsetState <ORAHOME>
            <user_mailid> <user_password> ");
            System.exit(1);
        }

        try
        {
            OESContext oesctx = new OESContext(
                                        DirectoryConstants.DS_CALLERTYPE_APP,
                                        false);

            //Authenticate to the directory
            if (oesctx.authenticate(null, args[0]) != true)
            {
                System.err.println("APP authentication failed");
                System.exit(1);
            }
            else
            {
                //Authenticate the user now
                OESContext clientCtx = new OESContext(DirectoryConstants.
                DS_CALLERTYPE_MAILUSER);
                clientCtx.authenticate(args[1], args[2], oesctx);
```

```
                DirectoryAccess access = new DirectoryAccess();
                System.out.println("Retrieving user state");

                Vector state = access.RetrieveAttribValueFromCache(clientCtx,
                "orclmailuserstate");

                if (state!= null)
                  System.out.println("User State is " + (String)state.elementAt(0));
                    //Callers can check if the state is "active" or "migrating"
                    and decide their action based on that.
            }

            System.gc();
        }
        catch (Exception e)
        {
            System.err.println("Error Occured " + e.getMessage());
            System.gc();
        }
    }
```

ルール管理 API

ルール管理 API は、サーバー・サイド・ルールの作成、アクセスおよび管理に使用できる Java クラスのセットです。ルールは Java オブジェクトとして表され、Oracle Internet Directory にユーザーの属性として永続的に保存できます。

この項では、次の項目について説明します。

- サーバー・サイド・ルール
- ルール・コンポーネント
- 認証
- ルールの可視性、アクティブ性、グループへの所属
- 外部条件
- メッセージ・テンプレート
- 自動応答の有効期間
- XML 表現

関連資料： ルール管理 API の詳細は、「Oracle Email API Reference」（Java Doc）を参照してください。

サーバー・サイド・ルール

メール・ルールは、特定のイベントが発生し、特定の条件が満たされたときに、ルールの所有者にかわって電子メール・メッセージに対して行われる潜在的なアクションです。ルールは、メール・サーバーに永続的に作成および格納できます。

次に、言葉で表されたルールの例を示します。

「電子メールが受信ボックスに届き、その件名に「Get paid to surf」というフレーズが含まれている場合、メッセージを削除します。」

この例は次のことを表します。

- イベントは、受信ボックスに電子メール・メッセージが着信することです。
- 条件は、件名に指定されたフレーズが存在することです。
- アクションは、メッセージの削除です。

各イベントは、メール・サーバーのライフ・サイクル中に発生する特定のメッセージの状態変化を表します。上の例では、イベントはメッセージの状態を `To be delivered` から `Delivered` に変更します。

条件は、関係演算や論理演算によってメッセージ属性をテストするブール式と似ています。この例の場合、件名がテスト対象のメール属性で、条件式は、属性に「Get paid to surf」が含まれるかどうかをテストする関係演算です。

オプションで、AND や OR などの論理演算子を使用して複数の条件を組み合せ、複合条件を作成できます。さらに、ブール値を返す外部ファンクションコールを条件として使用できます。

アクションは、メッセージの削除など、メッセージに対して行われる操作です。アクションは、メール・サーバー内から PL/SQL でコール可能な外部プロシージャの場合もあります。

ルール・コンポーネント

ルールは、個々のユーザーまたはユーザーのグループがまとまって所有します。したがって、ルールを所有する最上位レベルのエンティティは、メール・ユーザー、ドメインまたは電子メール・システム全体のいずれかで、複数のドメインが含まれることがあります。最上位レベルのエンティティをアカウントと呼びます。

新規メールの配信時、あるいはメッセージを開いたときなどの一連のイベントに対して、アカウントごとにルールを定義できます。各イベントはルール・リストと関連付けられ、アカウントには複数のルール・リスト（1 つのイベントに対して 1 つのルール・リスト）を指定できます。どのイベントについても、ルール・リストには、イベントの発生時に連続して実行されるルールのリストを含めることができます。

ルールは、1 つの条件とアクションのリストによって定義されます。条件のないルールは無条件のルールと呼ばれ、そのアクションは常に実行されます。

条件には、単純な条件と複雑な条件があります。たとえば、関係演算子を使用して属性をリテラル値と比較する条件は単純な条件です。複雑な条件では、複数のサブ条件を組み合せることができます。条件は、外部条件と呼ばれるユーザー定義プロシージャにすることもできます。セクション内条件と呼ばれる特別な条件もあります。この条件は、メッセージで内容に基づく検索を実行します。

ルールの条件が満たされた場合、アクションが実行されます。アクションは、「メッセージをフォルダに移動」や「メッセージを受信者に転送」などのルールのコマンドと、メッセージの移動先のフォルダ名やメッセージを転送する受信者のアドレスなどの関連パラメータによって定義されます。ユーザーに対するすべてのルールを Java オブジェクトで構成した後、`RuleParser` オブジェクトを使用して保存します。

認証

コール元がユーザーのルールにアクセスする前に、コール元を認証する必要があります。コール元は、`oracle.mail.OESContext` クラスを使用して、Oracle Internet Directory で認証する必要があります。認証後、`setAuthContext()` メソッドを使用して、トラステッド・セッションを表す `oracle.mail.OESContext` クラスのインスタンスを `RuleParser` ルール管理クラスに渡す必要があります。

例

```
RuleParser parser = new RuleParser();
parser.setAuthContext(oesctx);
```

検証

サーバーでルールを作成する前に、実行時に無効なルールが実行されないように、ルールの内容が検証されます。`RuleParser.setValidation()` メソッドを使用して検証を使用不可にできます。これは、不完全なルールを一時的に保存する場合に便利です。検証されていないルールも永続的に保存できますが、実行時に実行することはできません。

検証は、次の方法で使用不可にします。

```
parser.setValidation(false);
```

ルールの可視性、アクティブ性、グループへの所属

ルールには、次のように分類されるいくつかの属性があります。

- 可視性
- アクティブ性
- グループへの所属

可視性

ルールは表示または非表示にできます。非表示のルールは、通常のルールと同じように機能しますが、ユーザーには表示されません。ルールを実際に非表示にするかどうかは、コール元が決定します。API では、表示および非表示のルールを両方とも取得できます。

可視性は、`RuleType` クラスの `setVisible()` メソッドを使用して設定します。

アクティブ性

ルールはアクティブまたは非アクティブにできます。非アクティブなルールはサーバー上に存在しますが、実行時に実行されません。

アクティブ性は、`RuleType` クラスの `setActive()` メソッドを使用して設定します。

グループへの所属

ルールはグループに属することができます。同じグループに属するすべてのルールを、1 つのコールで取得、アクティブ化および使用不可にできます。

グループへの所属は、`RuleType` クラスの `setGroup()` メソッドを使用して設定します。

RuleType の例

```
RuleType rule_t = new RuleType();
rule_t.setVisible("no");
rule_t.setActive("no");
rule_t.setGroup("group1");
```

外部条件

外部条件は、次の書式の PL/SQL ファンクションです。

```
function <func_name> (p_sessionid in integer,
       p_msgobj in mail_message_obj) return integer;
```

セッション ID とメッセージ・オブジェクトを指定すると、ファンクションは条件が満たされたかどうかを示す数値を返します。

戻り値が 0 の場合、条件は満たされたとみなされ、戻り値が 0 以外の場合、条件は満たされなかったとみなされます。ルールで外部条件ファンクションを使用する場合、条件を有効にする前に、ユーザーが属するデータベース・サーバーに手動でロードする必要があります。

条件を外部条件として設定するには、ConditionType クラスの addProcCall() メソッドを使用します。

外部条件の例

```
ConditionType cond_t = new ConditionType();
cond_t.addProcCall("ext_func_name");
```

外部アクション

外部アクションは、次の書式の PL/SQL プロシージャです。

```
procedure <proc_name>(p_event in number,
p_sessionid in number,
p_msgobj in mail_message_obj,
p_param1 in varchar2,
p_param2 in varchar2,
p_status out number);The event ID parameter takes the following possible values:
es_rule.c_copy
es_rule.c_deliver
es_rule.c_expunge
es_rule.c_flagchange
```

プロシージャも、セッション ID、現在のメッセージ・オブジェクト、およびルールの作成時に設定する 2 つのユーザー定義パラメータを取ります。プロシージャの完了後、ステータス・パラメータに実行結果値が入ります。ステータスがゼロ値の場合は正常に実行されたことを示します。ステータスが正の数値の場合、正常に実行されなかったことを示します。

サーバー・サイド・ルールを使用して外部アクションを設定するには、ActionType クラスの addCommand() メソッドを使用し、addParameter() を 3 度コールします。プロシージャ名が最初のパラメータとして追加され、上の例の p_param1 と p_param2 が第 2 および第 3 のパラメータとして追加されます。

外部アクションの例

```
ActionType action_t = new ActionType();
action_t.addCommand("call");
action_t.addParameter("ext_proc_name");
action_t.addParameter("param1");
action_t.addParameter("param2");
```

メッセージ・テンプレート

一部のルールには、新規メッセージを返信または通知として生成するアクションが必要です。返信または通知は、ルールの内容にテンプレートとして格納できます。このテンプレートには、2 つのパーセント（%）記号で囲まれたパラメータ名で表される置換可能なパラメータが含まれます。

たとえば、「%rfc822date% に送信された件名 %rfc822subject% の電子メールは受信されました。」などの自動返信テンプレートを作成できます。ルール・エンジンによって返信メッセージが作成されると、`%rfc822subject%` および `%rfc822date%` 変数は、受信メッセージに含まれていた実際の件名および日付情報に置き換えられます。サポートされるパラメータのセットは、サポートされているメッセージ属性のセットと同じです。

> **関連資料：** `AttributeType` クラスの詳細は、「Oracle Email API Reference」（Java Doc）を参照してください。

メッセージ・テンプレートのテキストは、Notify、Reply、Replyall、Forward などのルール・アクションのパラメータ値として使用されます。

メッセージ・テンプレートの例：

```
ActionType action_t = new ActionType();
action_t.addCommand("notify");
action_t.addParameter("jdoe@acme.com");
action_t.addParameter("Message Alert");
action_t.addParameter("You have received email from %rfc822from% regarding
%rfc822subject%");
```

自動応答の有効期間

受信ボックスが自動応答メッセージでいっぱいになるのを防ぐために、特定の返信アクションに有効期間を設定できます。期間は日数で指定します。特定のメッセージ・テンプレートを使用して特定のアドレスに自動応答が送信されると、最初の返信の送信時刻から始まる有効期間の間、同じ返信が同じユーザーにもう一度送られることはありません。ルール・アクションの Reply および Replyall に値が必要です。

有効期間の例 :

```
ActionType action_t = new ActionType();
action_t.addCommand("reply");
action_t.addParameter("7");
action_t.addParameter("Message received");
action_t.addParameter("Your email regarding %rfc822subject% sent on %rfc822date% has
been received.");
```

XML 表現

ルール・データはシリアライズできます。つまり、XML を使用してプレーン・テキスト形式に変換できます。これで、アプリケーション間で簡単に転送したり、オフラインで格納できるようになります。実際、ルール API は、Oracle Internet Directory への格納に際し、内部で XML を書式として使用しています。ルール Java オブジェクトを XML テキストにフラット化するには、`Account` クラスから `print()` メソッドを使用します。

XML の例

```
XMLOutputStream out = new XMLOutputStream(System.out);
account.print(out);
out.writeNewLine();
out.flush();
```

次の例では、API を使用して単純なルールを作成する方法を示します。

```
import oracle.xml.classgen.InvalidContentException;
import oracle.xml.parser.v2.XMLOutputStream;
import java.util.*;
import java.io.*;
import oracle.mail.*;
import oracle.mail.sdk.rule.*;
import oracle.mail.sdk.ldap.*;
public class Demo {
    public static main (String args[]) throws Exception {
        // login to LDAP using Directory APIs
        OESContext appCtx = new OESContext(DirectoryConstants.DS_CALLERTYPE_APP);
        appCtx.uthenticate(null, "/your/local/oracle/home");

        // authenticate as a rule owner
        OESContext clientCtx = new OESContext(ESDSConstants.DS_CALLERTYPE_MAILUSER);
        clientCtx.authenticate("testuser1@oracle.com", null, appLogin);
        // set authentication context in RuleParser
        parser = new RuleParser();
        parser.setAuthContext(clientCtx);
        // first create the top level user account type object
        AccountType acnt_t = new AccountType();
```

```
// set ownerType, either system, domain or user (default)
acnt_t.setOwnerType("user");

// for system rules, this is the installation name in LDAP,
// such as "install1", for domain rules this is the domain
// name, such as "oracle.com", for user rules this is the
// fully qualified username, such as testuser1@oracle.com
acnt_t.setQualifiedName("testuser1@oracle.com");
// create a rulelist type object, set the event
RuleListType rlist_t = new RuleListType();
rlist_t.setEvent("deliver");

// create a rule type object
RuleType rule_t = new RuleType();
rule_t.setDescription("a new rule");
rule_t.setGroup("group1");

// create a condition type object
ConditionType cond_t = new ConditionType();

// create a simple attribute:
cond_t.addAttribute("rfc822subject");
// or create an attribute object with parameters:
//
// AttributeType attr_t = new AttributeType("xheader");
// attr_t.setParam("X-Priority");
// cond_t.addAttribute(attr_t);
// create a simple operator:
cond_t.addOperator("contains");
cond_t.addOperand("Hello");

// create an external condition
ConditionType cond_t2 = new ConditionType();
cond_t2.addProcCall("extern_cond");

// create a negation of disjunction of above two conditions
// (i.e. not (cond1 or cond2) )
ConditionType cond_t3 = new ConditionType();
cond_t3.setJunction("or");
cond_t3.setNegation("yes");
cond_t3.addCondition(cond_t);
cond_t3.addCondition(cond_t2);

// add the condition object to the rule type object
rule_t.addCondition(cond_t3);
```

```
            // create an action type object
            ActionType action_t = new ActionType();
            action_t.addCommand("moveto");
            action_t.addParameter("/testuser1/folder1");
            // add the action to the rule object
            rule_t.addAction(action_t);

            // create a second action object and add it in the rule type
            ActionType action2_t = new ActionType();
            action2_t.addCommand("call");
            action_t.addParameter("extern_action");
            action_t.addParameter("param1");
            action_t.addParameter("param2");
            rule_t.addAction(action2_t);

            // add the rule object in the rulelist type object
            rlist_t.addRule(rule_t);

            // add the rulelist object in the account type object
            acnt_t.addRulelist(rlist_t);
            // create an account object on based the type object
            Account acnt = new Account(acnt_t);
            parser.setValidation(true); // default
            parser.setRuleObjects(acnt);
        }
    }
```

携帯情報端末用フィルタおよびプロファイル API

携帯情報端末や低速リンクから電子メールに効率的にアクセスするには、フォルダに電子メールの小さなサブセットのみをダウンロードできるフィルタを設定する必要があります。ユーザーは、携帯情報端末用のフィルタおよびプロファイル機能を使用して、フォルダで検索条件を定義できます。検索条件を定義してフォルダを開くと、条件を満たすメッセージのみが選択され、クライアントに表示されます。ユーザーは、様々なメッセージのセットにアクセスするためのプロファイルを使用して、フォルダで複数の条件を定義することもできます。

次の携帯情報端末用フィルタおよびプロファイル・コンポーネントがあります。

- フィルタ： フォルダで定義する検索条件です。IMAP プロトコルで定義されたすべての検索条件をサポートします。これには、AND や OR で結合された複雑な条件も含まれます。
- 仮想フォルダ： 定義済のフィルタを持つフォルダ。クライアントがフォルダを開くと、フィルタ条件を満たすメッセージのみが表示されます。

- プロファイル： 仮想フォルダのセットに関連付けられた名前。ログイン時に使用され、サーバーに適用するフィルタを指定します。たとえば、ユーザーが INBOX フォルダで「緊急メッセージのみを表示」という条件の WP1 というプロファイルと、「送信者 *john* および *scott* からのメッセージを表示」という条件の WP2 という 2 番目のプロファイルを定義したとします。ユーザーが *username#wp1@domain_name* としてログインし、INBOX を選択すると、緊急としてマークされたメッセージだけが表示されます。また、ユーザーが *username#wp2@domain_name* としてログインし、INBOX を選択すると、送信者が *john* および *scott* のメッセージが表示されます。

  仮想フォルダをサポートするインタフェースは次のとおりです。

  - Java SDK: Java SDK の一部として、プロファイルとフィルタの作成および変更にプロシージャを使用できます。さらに、仮想フォルダにアクセスするための特別なログイン・フォーマットもサポートされています。
  - IMAP サーバー： 仮想フォルダにアクセスするための特別なログイン・フォーマットがサポートされています。これで、Outlook Express や Netscape Communicator などの標準ベースのクライアントから仮想フォルダにアクセスできます。

携帯情報端末用フィルタのリスト

次の例では、携帯情報端末用フィルタのリスト方法を示します。

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;

import javax.mail.search.*;

import oracle.mail.sdk.esmail.OracleEsProfile;
import oracle.mail.sdk.esmail.OracleEsFilter;
import oracle.mail.sdk.esmail.OracleStore;
public class ListFilter
{
        static String password = null;
        static String user = null;
        static String host = null;
        static int port = -1;
        static String mbox = "INBOX";
        static String root = null;
        static boolean recursive = false;
        static String pattern = "*";
        static boolean verbose = false;
        static String namespace = null;
```

```
public static void main (String argv[]) throws Exception
{
        for (int i = 0; i < argv.length; i++)
        {
                if (argv[i].equals("-U"))
                        user = argv[++i];
                else if (argv[i].equals("-P"))
                        password = argv[++i];
                else if (argv[i].equals("-D"))
                        host = argv[++i];
                else if (argv[i].equals("-I"))
                        port = Integer.parseInt(argv[++i]);
                else if (argv[i].equals("--")) {
                        i++;
                        break;
                }
                else if (argv[i].startsWith("-")) {
                        System.out.println(
                        "Usage: ListFilter [-D ldap_host] [-I ldap_port]
                        [-U user] [-P password]");
                        System.exit(1);
                }
                else {
                        break;
                }
        }

        Properties props = System.getProperties();
        Session session = Session.getDefaultInstance(props,null);
        session.setDebug(true);

        Store store = null;

        store = session.getStore("esmail");

        if (user != null && password != null && port != 0 && host != null){
        store.connect(host, port, user, password);

        OracleEsProfile wp2 = new OracleEsProfile(store, "wp2", "Wireless
        Profile 2");
        wp2.create();
        OracleEsProfile[] profiles = ((OracleStore)store).listProfile();
        System.out.println("Number of profiles = " + profiles.length);

        for (int i = 0; i < profiles.length; i++)
        {
           System.out.println("PROFILE " + (i+1));
```

```
                    System.out.println("profile name = " + profiles[i].getName());
                    System.out.println("profile description = " +
                    profiles[i].getDescription()+ "\n");
                    System.out.println("List the filters:");
                    OracleEsFilter[] filtersList = profiles[i].listFilters();
                    System.out.println("Number of filters : " + filtersList.length);
                    for (int j = 0; j < filtersList.length; j++)
                    {
                       System.out.println("filter folder = " +
                       filtersList[j].getFolder().getFullName());
                       System.out.println("filter description = " +
                       filtersList[j].getDescription());
                       SearchTerm st = filtersList[j].getCriteria();
                       if (st instanceof FromTerm)
                           System.out.println("from term");
                    }
                  }

               }
                  store.close();
              }
}
```

プロファイルへのフィルタの追加

次の例では、プロファイルに携帯情報端末用フィルタを追加する方法を示します。

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
import java.io.*;

import javax.mail.search.*;
import javax.mail.Address;
import javax.mail.internet.InternetAddress;

import oracle.mail.sdk.esmail.OracleEsProfile;
import oracle.mail.sdk.esmail.OracleEsFilter;
import oracle.mail.sdk.esmail.OracleStore;
public class AddFilter
{
        static String password = null;
        static String user = null;
        static String host = null;
        static int port = -1;
        static String mbox = "INBOX";
        static String root = null;
```

```
static boolean recursive = false;
static String pattern = "*";
static boolean verbose = false;
static String namespace = null;

public static void main (String argv[]) throws Exception
{
        for (int i = 0; i < argv.length; i++)
        {
                if (argv[i].equals("-U"))
                        user = argv[++i];
                else if (argv[i].equals("-P"))
                        password = argv[++i];
                else if (argv[i].equals("-D"))
                        host = argv[++i];
                else if (argv[i].equals("-I"))
                        port = Integer.parseInt(argv[++i]);
                else if (argv[i].equals("--")) {
                        i++;
                        break;
                }
                else if (argv[i].startsWith("-")) {
                        System.out.println(
                        "Usage: AddFilter [-D ldap_host] [-I ldap_port]
                        [-U user] [-P password]");
                        System.exit(1);
                }
                else {
                        break;
                }
        }

        Properties props = System.getProperties();
        Session session = Session.getDefaultInstance(props,null);
        session.setDebug(false);

        Store store = null;

        store = session.getStore("esmail");

        if (user != null && password != null && port != 0 && host != null){
        store.connect(host, port, user, password);

        OracleEsProfile wp2 = new OracleEsProfile(store, "wp2", "Wireless
        Profile 2");
        wp2.create();
```

```
            OracleEsProfile[] profiles = ((OracleStore)store).listProfile();
            System.out.println("Number of profiles = " + profiles.length);

            for (int i = 0; i < profiles.length; i++)
            {
               System.out.println("PROFILE " + (i+1));
               System.out.println("profile name = " + profiles[i].getName());
               System.out.println("profile description = " +
                profiles[i].getDescription()+ "\n");
            }
            Folder folder = store.getFolder("xyz");
            OracleEsFilter filter = new OracleEsFilter(folder, "F2", new
            FromTerm(new InternetAddress("tuser1@us.oracle.com")));
            profiles[0].addFilter(filter);
         }
            store.close();
        }
}
```

3

SMTP スキャナ・インタフェース API

この章では、カスタマイズされたクライアントの作成、アプリケーションの統合、特定の拡張のサポートで使用できる Oracle Email SMTP スキャナ・インタフェース API について説明します。

次の項目について説明します。

概要

SMTP サーバーは、フィルタリングおよびスキャンのためのユーザー定義のプロシージャと統合することができます。これらのプロシージャは、C 言語関数として実装され、サーバーにリンクされます。各フィルタおよびスキャナは、サーバーが処理中の異なる段階でコールする関数をセットとして実装されなければなりません。各スキャナには、メッセージ・ヘッダーとデータを取得するためにコールできるコールバック関数のセットがあります。

フォーマットおよびエントリ

スキャナ・インタフェースのリストを指定する LDAP サーバー・パラメータ内のフォーマットおよび一般的なエントリです。

shared-library-path, *when-to-call*, [internal | *host*:*port*], function set: (initialization, register callback, scan message, send, receive, close),*scanner_flags*,*system_flags*

- *shared-library-path* には、C 言語共有ライブラリのフルパスを指定します。サーバー起動時にロードされます。
- *when-to-call* には、C 言語インタフェースをコールするタイミングを指定します。次の値が使用できます。
  - ENV: メッセージ・エンベロープの受信後
  - DATA: 完全なメッセージの受信後およびローカル配信前
  - RELAY: メッセージのリレー直前
  - NEVER: インタフェースの呼出しを無効にする場合
- *host* および *port* には、ウィルス・スキャナが監視するホストおよびポートを指定します。
- Internal は、フィルタリングのための外部エンティティとの接続を行わないスキャナの場合に指定します。
- *scanner_flag* には、スキャナが初期化関数に渡す必要のあるフラグを指定します。
- *system_flags* には、MTA 定義フラグを指定します。現在、定義できる値は、repairmsg=0/1 のみです。0 を設定すると、メッセージに不具合が発生した場合、スキャナ・インタフェースによりメッセージは廃棄されます。1 を設定すると、メッセージは修復されます。

例

```
/work/orahome/lib/libsym.so,DATA,host9999:6000,(essym_init,essym_registercb,
essym_scanmsg, essym_send, essym_recv, essym_close),0, repair_msg=1
```

スキャナ・インタフェース API

次の関数は、各スキャナによって実装する必要があります。

- vgctx: ウィルス・スキャナ・インタフェースのグローバル・コンテキスト。スキャナ API により初期化および管理されるグローバル・コンテキストです。各スキャナ・インタフェース・コールに渡されます。
- vlctx: 各メッセージのためのウィルス・スキャナ・インタフェースのローカル・コンテキスト。各メッセージで初期化および使用されます。
- smtplctx: SMTP サーバーのローカル・コンテキスト。サーバーの内部で使用されます。
- msgin: メッセージを含むバッファ。このリリースでは使用しません。

Initialization()

パラメータ : `void **vgctx, char *host, sb4 port, char *system_flags, char *scanner_flags`

戻り値 : `int`

説明 :

サーバー起動時に一度だけコールされます。サーバー存続期間中に一度しかコールされません。スキャナを使用するために、グローバル・コンテキストを作成する最小手順でスキャナの初期化を実行します（次のフレームワークでの使用のために vgctx の void ポインタを戻します）。

- ホストおよびポートをストアし、オプションで次に使用するためのウィルス・スキャナへの接続を確立します。
- system_flags パラメータに repairmsg=1 が設定されていると、メッセージの修復が実行されます。
- Scanner_flags はスキャナ定義のフラグのセットです。

register_callback()

パラメータ : `void *vgctx, int (*cb)(void*, void*,void*), unsigned int flags)`

戻り値 : `int`

説明 :

スキャナがメッセージ情報を取得するためにコールできる関数のセットをサーバーが提供します。これらの関数名は事前定義されず、サーバーによってこの関数に渡されます。`initialization()` をコール後、サーバーが、メッセージ・データを取得するために使用可能な各サーバー関数のために、この関数をコールします。次にスキャン・メッセージで使用するために、コールバック・ルーティン cb をストアします。

フラグ・パラメータは、次の値でコールバックの種類を指定します。

- 0: メッセージの取得および送信
- 1: メッセージの受信およびストア
- 2: エンベロープの取得
- 3: メッセージ・サイズの取得
- 4: メッセージ ID の取得
- 5: メモリーの解放
- 6: スキャナ・バージョンの定義の設定

scan_message()

パラメータ : `(void *vgctx, void *smtplctx, char *msgin, char *errmsg)`

戻り値 : `int`

説明 :

- 各メッセージのためのサーバーがこの関数をコールします。メッセージのスキャンおよびウィルス・スキャナからの結果の取得に必要な処理を実行します。メッセージ・ヘッダおよび本文の `retrieve-send` および `receive-store` のためにコールバックを使用します。
- `msgin` は、メモリにあるメッセージの全体を渡すために使用されます。このため、`retrieve-send` のコールバックは、メール・ストアからのメッセージの取得には使用できません。
- エラーが発生した場合、`errmsg` にはエラー・メッセージが移入されます。`errmsg` のバッファは、サーバー内に 512 バイト割り当てられています。

戻り値 :

- 0: 成功
- 1: 失敗
- 2: 修復
- 3、1 以上 : エラー

send()

パラメータ : (void *vlctx, char *buf, int buflen)

戻り値 : int

説明 :

- スキャナはメッセージ本文を取得するために retrieve および send message のコールバックをコールします。スキャナへメッセージを提供するために、このコールバックは内部的に複数回、Send() 関数をコールします。
- buflen は、buf で指定したバッファのサイズです。
- 戻り値 : 送信エラーでは、1 以上。

recv()

パラメータ : (void *vlctx, char *buf, int *buflen)

戻り値 : int

説明 :

- スキャナに変更されたメッセージを戻すために receive および store message のコールバックをスキャナがコールします。メッセージ・データを取得するために、このコールバックは内部的に recv() 関数をコールします。
- buflen は、recv() 関数が buf で指定したバッファ内に確保するバイト数です。EOF により *buflen で指定したバイト数より少ない場合、buflen は実際のバイト数に更新されます。
- 戻り値 : 受信エラーでは、1 以上。

close()

パラメータ : (void *vgctx);

戻り値 : void

説明 : この関数は、サーバー終了時に一度だけコールされます。接続の終了、メモリーの解放などの必要なクリーンアップ処理が実行されます。

コールバック・ルーティンの説明

これらの関数は、メッセージ・データを取得するために使用されます。すべてのコールバック・ルーティンのシグニチャは汎用的なものです。例 : int (*cb) (void *, void *, void *)。パラメータの値は、各コールバックのために次のものがあります。

Send Message

スキャナのためにメッセージを取得します。

パラメータ : (void *vgctx, void *vlctx, void *smtplctx)

Receive Message

スキャナから変更されたメッセージを読み込みます。

パラメータ : (void *vgctx, void *vlctx, void *smtplctx)

Get Envelope

メッセージ・エンベロープを取得します。

パラメータ : (void *vgctx, void *output, void *smtplctx)

output は、host=hostname|mailfrom=sender info|rcptto=(rcpt1,rcpt2,...) フォーマットのエンベロープ情報を持つ char** 型で、フレームワーク内でメモリーが割り当てられます。

Get Message Size

メッセージ・サイズを取得します。

パラメータ : (void *vgctx, void *output, void *smtplctx)

output は、メッセージ・サイズを含む結果の整数へのポインタです。

Free Memory

Get Envelope コールバックの出力パラメータを解放します。

パラメータ : (void *vgctx, void *input, void *smtplctx)

input は、解放されるバッファへのポインタです。このバッファは、フレームワークによって前もって割り当てておく必要があります。

Set Version Definition
メッセージをスキャンするために使用するスキャナのバージョンを設定します。

パラメータ：(void *vgctx, void *input, void *smtplctx)

input は、ウィルス・スキャナのためのバージョン定義を含む文字列です。ウィルス・スキャナからバージョン定義情報を使用できる場合、このルーティンは、スキャナ API によってコールされる必要があります。

索引

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

め

ゆ

る

れ