



LYRIS ListManager

Lyris ListManager 7 Programmer's Guide

<http://www.synaptive.com>
info@synaptive.com

Table of Contents

Table of Contents	1
Introduction to ListManager	2
Direct Database File Access and ListManager	2
About the queries	3
Adding a member	3
How many users are on list xxx	3
Introduction to the API	3
Mail commands	3
Direct database access	3
Lyris Command Protocol (LCP)	4
TclPort	4
Introduction to TclPort	4
TclPort Database Functions	4
TclPort Security Functions	7
TclPort Mail Queue Functions	11
Introduction to the TclPort Terminal	11
The ListManager Tables	11
Future Direction	12
"people_" table	13
"respond_" table	13
"phrases_" table	14
"log_" table	15
"domainconnectionlimits_" table	15
"dnsbypass_" table	16
"bannedmembers_" table	16
"config_" table	17
"sites_" table	18
"lists_" table	19
"topics_" table	25
"mimeencodings_" table	26
"mimecharsets_" table	26
"messagetypes_" table	27
"docs_" table	27
"doctypes_" table	28
"listdocs_" table	28
"sitedocs_" table	28
"docparts_" table	29
"mimetypetoext_" table	29
"mimetypes_" table	29
"mimeexts_" table	30
"clickgroups_" table	30
"clickstreamdata_" table	30
"urls_" table	31
"clicktracking_" table	31
"usercolumninfo_" table	31
"axistypes_" table	32
"chartdescription_" table	32
"lyrMetricEvents" table	33
"lyrPerformanceMetrics" table	33
"lyrLookupCompletionStatus" table	33
"lyrCompletedRecips" table	33
"inmail_" table	34
"lyrActiveRecips" table	36
"moderate_" table	37
"outmail_" table	38
"members_" table	39
"listmessagesmapping_" table	41
"messages_" table	42
"lyrPermissionGroups" table	43
"lyrPermissionGuiAttribs" table	43
"subsets_" table	43
"translatekeys_" table	45
"translatelang_" table	45
"translatevalues_" table	45
"referrals_" table	46
"messengerrelationship_" table	46
"lyrTaskScheduler" table	46
"lyrTaskDescription" table	47
"lyrSurveyResponseAnswer" table	47
"lyrSurveyResponse" table	48
"lyrSurveyQuestions" table	48
"lyrSurveyAnswers" table	48
"lyrSurveyAnswerCollections" table	49
"lyrWebDocs" table	49
"lyrLookupWebDocTypes" table	50

Introduction to ListManager

Lyris ListManager is a powerful, fast, and easily extensible tool to help manage your email distribution. For programmers, ListManager has a number of methods to customize and manipulate the behavior of the product.

ListManager uses powerful relational databases, such as Microsoft SQL Server or Oracle to store its data. The table layout used by ListManager is provided in this document, so users should feel free to write whatever extensions or custom reports that meet their needs. With this information programmers can directly insert data, such as new members, in whatever programming language or environment that best meets the company's needs.

Beyond the exposed database design, the ListManager environment can be altered programmatically in a number of ways. There are pre-defined "hooks" in the program to run scripts at certain key times, and certain functionality is designed to be flexible, such as the automatic phrases content checks.

Lyris ListManager is the only product on the market that allows a programmer to create scripts that can be run to customize a message for each recipient, "on the fly". With this ability a programmer can create a script using the Tcl scripting language that customizes the contents of a message, one time, if desired, or to make it unique for each recipient. Ad banners can be rotated, flight schedules inserted, nearly anything that can be imagined.

The source code for key portions of the included web interface are provided with the product at <http://yourlistmanagerserver/docs/api/index.html>. This allows a programmer to customize the interface to make it fit within company look-and-feel standards or to augment the behavior for a custom subscribe form, for example.

ListManager has an exposed Application Programming Interface (API) called TclPort for web programmers and others who are interested in augmenting or changing the behavior of ListManager. Using TclPort, a programmer can create powerful Tcl scripts that can be run on the ListManager server, manipulating data in the SQL database or in program variables.

Direct Database File Access and ListManager

It is possible to directly manipulate the ListManager database. Much of the data in the database is related to other data, so changes to one field might require a change to another field in another table. Making modifications without understanding the data dependencies and field definitions can lead to unexpected results. It is highly recommended that users test their scripts on a private test list before running them on public lists.

Changing the data in tables while ListManager is running is a little dangerous but should generally work. You need to understand the database design well enough to know if changes made to the tables will have an immediate effect or not. In some cases, ListManager is regularly polling a table for changes, but in other cases, it won't pick up a change until the system is restarted.

Programmers should feel free to work with the members_ table. Many companies want to be able to add or change the membership information in the members_ table dynamically, and doing so generally should not cause problems. Programmers are encouraged to import their membership data directly into the database server using the native tools provided by the database. Using these tools results in import speeds of millions of records per hour, which is faster than any other method available through ListManager.

While ListManager is expecting users to add columns to the members_ table, it will very likely cause errors if existing columns/indexes are changed, or the schema is modified in other tables. Developers interested in making demographic data available for mail merging or for the purpose of determining whom to send mail are encouraged to add columns to the members_ table or to a new table. ListManager has the easy ability to pull data from user-provided tables during the mail merge process.

All tables are designed by Lyris for the purpose of making ListManager as fast and powerful as it is today. As the technology demands changes, any of the tables used by ListManager today may be changed in subsequent versions of ListManager, so application programmers need to keep this in mind. Lyris offers no guarantees that the schema used for one version of ListManager will be the same for another version.

About the queries

This section provides some common queries that could be useful for programmers, either as starting points for more advanced queries, or as they exist today integrated into web pages or other places where this data could be useful. There are many more complicated queries that could generate reports or answer more specific questions, but these queries are designed to be fairly simple and generic queries useful by a large number of clients.

Adding a member

The members_ table stores a row for each list that the member has joined, so the same email address will appear multiple times in this table. Most columns have defaults, so the minimum number of columns can be provided for an insert. You **must** split the user name and domain portion of the email address prior to inserting a row into this table. The UserNameLC_ field stores the user name in lower case, and the domain should always be in lower case. The EmailAddr_ field stores the email address in mixed cases, and **must** be the same as the UserNameLC_ + @ + Domain_, ignoring case, or ListManager will not work properly.

Records can be inserted into the members_ table with a 'needs-' status, such as needsconfirm. ListManager regularly polls new records into the members_ table to see if any of the new members need a hello, confirm, or goodbye document sent to them. The check is only done regularly for new members, so updating an existing member to 'needs-hello', for example, will not cause an immediate document send. Instead, during a nightly check this status change will be noticed and the document sent.

Members can be added or marked unsubscribed at any time in ListManager. Postings which have been processed to the point that sending has started will not reflect changes made in subscriptions after that point. So if a posting is created to ten people and it is processed to become an outmail record, when user number eleven is added to the list he/she is not automatically sent the posting, even if the job has not yet completed sending to all recipients.

The MemberID_ field is an auto-numbering field, so should never be provided for insert statements.

```
insert into members_ ( EmailAddr_, UserNameLC_, Domain_, List_, MemberType_ ) values
( 'GeorgeBush@example.com', 'georgebush', 'example.com', 'terrorist-watchnews',
'needs-confirm' )
```

How many users are on list xxx

To determine how many users are subscribed to a particular list, simply query theMembers_ table:

```
select count(*) from members_ where list_ = 'jazztalk' and MemberType_ = 'normal'
```

If you want to know the breakdown of members by type, on a list:

```
select MemberType_, count(*) from members_ where list_ = 'jazztalk' group by MemberType_
```

Introduction to the API

A programmer has several choices about how to programmatically solve problems with Lyris ListManager. One technology may be appropriate for one type of customization, where another technology may work better for a different customization. Here are some of the choices:

Mail commands

Many powerful commands are available as email commands. Members can be subscribed, unsubscribed, and their setting altered through email commands. Data about the membership and other reports can all be requested via email. If your need can be filled with these simple commands you may want to create emails to ListManager rather than using one of the more complicated programming options.

Direct database access

All the tables in Lyris ListManager are accessible through the SQL language. Data may be inserted, deleted,

or modified in these tables directly. For some tasks, such as member creation, it may be easiest to use the programming language and environment you are comfortable with to access the database server using SQL to insert members. However, if you are uncertain about the use of the data or the relationship of the data to program functionality it is probably best to avoid making any direct database modifications.

Lyris Command Protocol (LCP)

The LCP interface has been removed and is no longer available. Application programmers should use direct SQL, TclPort, or email commands.

TclPort

The TclPort interface allows Tcl scripts to be run on the ListManager server. This interface allows great flexibility since the script is dynamically interpreted at execution time. It is also quite fast, even though it is flexible, and allows Lyris developers to code performance-critical sections in C++ and efficiently call them from Tcl. The ListManager web interface communicates with the ListManager server entirely using TclPort, and every function that the web interface uses is available to application programmers.

For information on the Web based API, please see Utilities: Other: Programming API.

Introduction to TclPort

TclPort is a Remote Procedure Call (RPC) interface to ListManager. Since the web interface is implemented in Tcl, it is natural to use the same types of Tcl command calling conventions to execute commands on the ListManager server. By having a Tcl interpreter in the back-end, it is easy to process complicated data structures passed in or out through TclPort. It is not generally used as a normal Tcl interpreter but rather, the Tcl C library functions are used to do the processing.

TclPort can only be accessed through the `tclport.tcl` library which handles all connection, argument and result processing. The protocol was designed in a way that makes it impractical to use directly via Telnet. However, there is a web-based interface to TclPort that handles all argument and result processing. See Introduction to the TclPort Terminal for more information. The ListManager web interface relies on TclPort for all database interaction. Additionally, there are other commands that act directly on the ListManager back-end.

To use TclPort in the web interface, simply pass the name of the command and its arguments to the "tclport" comment in the web interface. For example:

```
set lists [tclport get_admin_privileges "list" "email@address" "p455w0rd"]
```

In this case, the results are a list of mailing list names which are stored in the `lists` variable.

TclPort Database Functions

sqlinsert

A procedure for doing an SQL INSERT and optionally returning the identity column value.

Arguments:

`tablename` - the name of the table to insert data into.

`data` - a Tcl list of columnname-value pairs that define the data to be inserted.

`get_identity` - a boolean value that determines whether or not the function should retrieve and return the identity value.

Returns:

If the `get_identity` argument is set to 1 then the identity column value for the new record will be returned. Otherwise, 0 will be returned.

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the `catch` command.

sqlinsertmultiple

A procedure for doing multiple SQL INSERT statements and optionally returning the identity column values.

Arguments:

tablename - the name of the table to insert data into.

data - a Tcl list of lists of columnname-value pairs that define the data to be inserted.

get_identity - a boolean value that determines whether or not the function should retrieve and return the identity value.

Returns:

If the get_identity argument is set to 1 then the identity column value for the new record will be returned. Otherwise, 0 will be returned.

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

sqlupdate

A procedure for doing an SQL UPDATE.

Arguments:

tablename - the name of the table to insert data into.

data - a Tcl list of columnname-value pairs that define the data to be inserted.

where - the where clause that identifies the record/s to update.

Returns:

Nothing.

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

sqlupdatemultiple

A procedure for doing an SQL UPDATE

Arguments:

tablename - the name of the table to insert data into

data - a Tcl list of lists of columnname-value pairs that define the data to be inserted where - the where clause that identifies the record/s to update

Returns:

Nothing.

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

sqlexecute

A procedure for executing an arbitrary SQL statement and returning a single result set.

Arguments:

statement - the SQL statement to execute.

limit - the maximum number of rows to return (optional - 0=unlimited).

offset - an offset index into the result set (optional - default 0).

Returns:

A list of data that represents the results of the query elements:

- the number of rows of data returned
- the number of columns of data returned
- a Tcl list of the returned data column names

- results by column then row. The first character of each result value is either 0 or 1 where 1 means that the value is NULL

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

sqlselect

A procedure for doing an SQL SELECT.

Arguments:

column_list - a list of columns to select.

tablename - the table to select from.

whereclause - the SQL WHERE clause (can be empty).

limit - the maximum number of rows to return (optional - default 0=unlimited).

offset - the number of rows to skip (optional - default 0).

Returns:

A list of data that represents the results of the query elements:

- the number of rows of data returned
- the number of columns of data returned
- a Tcl list of the returned data column names
- results by column then row.

The first character of each result value is either 0 or 1 where 1 means that the value is NULL

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

sqlcopy

A procedure for copying one row of data to another.

Arguments:

tablename - the name of the table to copy data within.

override_data - an associative array of column names and values to override in the

copy. where - a where clause that determines the records that are to be copied.

get_identity - set to 1 if you want the identity value of the last inserted value (optional).

exclude_columns - a list of columns to exclude from the copy (optional).

Returns:

if get_identity was set to 1, this command returns the identity value of the last row that was inserted. Otherwise, it returns 0.

Errors:

Any errors that occur are handled as Tcl errors and can be caught with the catch command.

tableinfo

A procedure for obtaining information about the ListManager database tables.

Arguments:

tablename - the name of the table.

Returns:

a list of the following values:

- the name of the table as returned by the database server
- the name of the identity column if it exists, otherwise, blank
- a list of the names of column names that make up a unique key into the table if they exist and an identity column was not returned, otherwise, an empty list

- a list of the names of column attributes that are being returned
- a list of lists of column attributes

membertableinfo

A procedure for obtaining information about the ListManager members table.

Arguments:

None.

Returns:

a list of the following values:

- members table name (may include the database name)
- email address column name
- member ID column name
- just the table name part of the members table name specification
- just the database name part of the members table name specification (may be blank)

TclPort Security Functions

license_capabilities

Returns information about the current license capabilities.

Arguments:

None.

Returns:

A list of key value pairs of license features and values.

license_setting

Returns information about a specific license code setting.

Arguments:

key - the name of the license code setting as returned by license_capabilities.

Returns:

the value for the specified key

refresh_license_capabilities

Forces ListManager to retrieve the current license capabilities from the license server.

Arguments:

"-background" - an optional argument that specifies that the procedure should return immediately and run the activation code refresh in the background.

Returns:

Returns a code that indicates the result of the activation code change

-1 = the activation code was changed

0 = the activation code wasn't changed

1 = license capabilities could not be updated due to a communications failure

2 = license capabilities could not be updated because your new activation code could not be retrieved.

3 = license capabilities could not be updated because your serial code is already registered for another hardware ID.

4 = license capabilities could not be updated because an invalid activation code was returned

get_server_id

Returns the server ID which is used when obtaining a new activation code.

Arguments:

None.

Returns:

The server ID.

cookie_get

Retrieves a string from the ListManager server memory which was previously stored with `cookie_get`.

Arguments:

key - the key used to retrieve the value.

Returns:

the value that was stored with the specified string.

cookie_put

Stores a string in the ListManager server memory which can be retrieved using the supplied key.

Arguments:

key - the key used to store the value.

value - the value to be stored.

Returns:

Nothing.

cookie_clear

Removes the data which is associated with the specified key.

Arguments:

key - the key used to store the value

Returns:

Nothing.

is_server_admin

Checks to see if the specified user is a server administrator.

Arguments:

username

password

Returns:

1 if the user is a server admin or 0 if they aren't.

is_site_admin

Checks to see if the specified user is a site administrator.

Arguments:

username

password

Returns:

1 if the user is a site admin or 0 if they aren't.

get_admin_privileges

Returns a list of list names or site names that the specified user is an admin of.

Arguments:

[list|site]
username
password

Returns:

Returns a list of list names or site names that the specified user is an admin of.

is_allowed_to_admin_this_site

Determines whether or not this user is allowed to admin the specified site.

Arguments:

sitename
emailaddr/username
password

Returns:

1 if the user is a site admin of the specified site or 0 if they aren't.

is_allowed_to_admin_this_list

Determines whether or not this user is allowed to admin the specified list.

Arguments:

sitename
emailaddr/username
password

Returns:

1 if the user is an admin of the specified list or 0 if they aren't.

is_allowed_to_admin_this_person

Determines whether or not this user is allowed to admin the specified person.

Arguments:

person name or email address
login name or email address
password

Returns:

1 if this login has admin rights for the specified person and 0 if they don't.

is_allowed_to_admin_this_(object)_id

Includes:

- is_allowed_to_admin_this_list_id
- is_allowed_to_admin_this_respond_id
- is_allowed_to_admin_this_phrase_id
- is_allowed_to_admin_this_doc_id
- is_allowed_to_admin_this_member_id
- is_allowed_to_admin_this_inmail_id
- is_allowed_to_admin_this_outmail_id
- is_allowed_to_admin_this_messages_id
- is_allowed_to_admin_this_moderate_id
- is_allowed_to_admin_this_subset_id

Determines whether or not this user is allowed to admin the specified object.

Arguments:

object ID
login name or email address
password

Returns:

1 if this login has admin rights for the specified object and 0 if they don't.

TclPort Server Functions

stats_all
stats_mail_send
get_machine_name
get_helo
refresh_dependencies

stats_all

Returns statistics about ListManager performance and resource usage.

Arguments:

None.

Returns:

A string.

stats_mail_send

Returns information about ListManager send speeds.

Arguments:

None.

Returns:

A stringstats_mail_send.

get_machine_name

Returns the name of the ListManager host machine which is used to store and retrieve configuration data from the config table.

Arguments:

None.

Returns:

A string.

get_helo

Returns the string used as the HELO command in SMTP transactions. Includes the "HELO " prefix followed by the domain of the sending machine as defined by the IP address used for sending.

Arguments:

None.

Returns:

A string.

refresh_dependencies

Refreshes any internal dependencies on configuration table settings

Arguments:

None.

Returns:

A string.

TclPort Mail Queue Functions**moderate_auto_approve_messages**

Approves any moderated messages that are scheduled for auto-approval before the current time.

Arguments:

None.

Returns:

A list of ID's for the messages that were approved.

check_for_new_mail

Forces an immediate check for new inmail.

Arguments:

None.

Returns:

Nothing.

Introduction to the TclPort Terminal

The TclPort Terminal is an address that programmers can use to test scripts or get results in real-time. This address gives a "terminal" where scripts can be run, either on the ListManager web interface, or on the ListManager server. This terminal can be a useful place to test scripts before actually deploying them in customer applications.

The address for the TclPort Terminal is:

`http://(your server)/utilities/debug/terminal.tml`

Only those specific IP address configured as TclPort hosts can access this terminal, so network administrators need not worry about security issues.

The ListManager Tables

All the ListManager tables are described in the following sections. Tables are broken down by their general function, to help with organization.

Tables are generally defined using Microsoft SQL Server keywords, although these tables work across all ListManager supported databases. The type BOOL listed in this document maps to a char(1) with a check constraint in all platforms. Here is a mapping of MSSQL keywords and their equivalent for other vendors, where the same type is not used for all vendors:

MSSQL column	MSSQL description	Used for Oracle	Oracle description	Used for Postgres
tinyint	A one byte value	int	A four byte value	int2
smallint	A two byte value	int	A four byte value	int2
int	A four byte value	int	A four byte value	int
bigint	An eight byte value	numeric(22, 0)	Up to 22 digits of	number(22, 0) *

			precision	
numeric	A numeric value numeric	numeric	A numeric value	numeric
smalldatetime	Time and date with accuracy to one minute	date	An 8-byte value with second precision	timestamp
datetime	Time and date with subsecond accuracy	date	An 8-byte value with second precision	timestamp
text	An unbound character column	clob	An unbound character column	text

* Postgres 7.3.x supports an 8-byte int value, but does not use indexes for these values under normal circumstances. Our understanding is that this is something that will be changed in future releases of Postgres. Until that time, ListManager will not use 8-byte ints for Postgres for columns that could be indexed. That means that some tables that use sequences will be limited to a 4-byte number of rows.

Additionally, Microsoft SQL Server supports a data column modifier called an "Identity" which can be added to numeric fields. This is an auto-numbering field that automatically populates the next incrementing value. Both Oracle and Postgres implement this with a separate "sequence" table. For ListManager, a sequence table is created with the base table name and "_seq", so "docs_seq", for example. With Postgres the next value is obtained from this sequence table via a default rule, where Oracle requires a 'before' trigger.

Previous versions of table and column names all ended with a trailing underbar, like 'lists_'. This was the legacy method to help avoid naming conflicts with other tables or objects in the database. Future versions of ListManager will drop the trailing underbars and will prefix all ListManager tables with 'lyr', like 'lyrWebDocs'.

Additional tables may be added to the database/tablespace holding the ListManager tables. As long as the names do not conflict there should be no problem with this.

ListManager will create the indexes it needs. Most tables have primary keys, and all tables that will benefit from indexes have them. Tables which have unique indexes on them rely on the uniqueness of the index, so they shouldn't be changed. Considerable effort is put towards finding the best mix of indexes for the different vendors and the different common usages of ListManager. While it may be possible to optimize the performance for reads for some tables by adding additional indexes, write performance will suffer. Users are free to add indexes if desired, but any changes to the structure of the tables may make future upgrades impossible.

The tables involved in mail sending (the mail queue) are the most active, for the common use of ListManager. These include lyrActiveRecips, lyrCompletedRecips, and inmail_ and outmail_. If message searching is enabled the messages_ table will store a copy of the message and will therefore grow and be somewhat active. Along with messages_, there are two tables that enable archives to be searched: wordmessage_ and uniquenesswords_. The members_ table is also one of the largest tables in ListManager.

Future Direction

The database tables have changes some as the product has grown from supporting only FoxPro to supporting standard client-server databases such as Microsoft SQL Server and Oracle. Lyris ListManager has some new tables and table changes. These type of changes are likely to continue.

It is likely that ListManager will continue to migrate the database design towards a more client-server architecture. The number of tables should increase, and the "width" of some tables will be reduced. More lookup and referential tables will be used to increase the speed and reliability of the design. In future versions, all text blobs that store one-to-many results should be replaced with table lookups.

In ListManager 7.8 the outgoing portion of the database mail queue was redesigned. It should be expected that many other tables will be altered in coming versions of ListManager.

The basic functionality of ListManager should not change, but the way in which some of the data is stored will change. If you write custom SQL statements to the current ListManager design you need to be prepared to change these as new versions of the software are released. Hopefully these changes will be simple, such as making a single-table lookup into a multi-table lookup using a join, but it could be more complicated.

For more information on ListManager tables, see About ListManager tables.

"people_" table

The people_ table hold server and site administrators, as well as other significant 'peopl'e to the server operation. A person can be listed in this table and not be a server or site admin because they may be a contact person for a list. The sites that an admin has rights over is listed as vertical bar (|) separated list in the WhatSites_ column. If a site admin needs to support more sites than can fit into this column a second people_ table entry should be created, or the person should be elevated to a server admin. This design is likely to change in the future.

Primary key:	Name_
Comment_	Comments about this person (text, NULL)
EmailAddr_	Person's email address (varchar(100), NULL)
FaxNum_	Person's fax number (varchar(50), NULL)
Name_	Unique name of the person (varchar(50), NOT NULL)
Password_	Password for this person (not needed if not admin) (varchar(50), NULL)
PostAddr_	Person's postal address (varchar(238), NULL)
ServerAdm_	Is this person a server admin? (bool)
Site_	Name of site this person belongs to. May be 'all', indicating that the sites are stored in WhatSites_. (varchar(50), NOT NULL)
SiteAdmin_	Is this person a site administrator? (bool)
TelNum_	Person's telephone number (varchar(50), NULL)
WhatSites_	What sites can this person administer. Site names in a string separated by (for site admin only) (varchar(238), NULL)

"respond_" table

The respond_ table holds information for autoresponders - addresses that, when mailed to, automatically send back an email document.

Primary key:	RespondID_
DocTitle_	What document should be mailed in response to getting email at this address (varchar(200), NOT NULL)
EmailAddr_	Email address that should produce an automatic response message, such as salesinfo@acme.com (varchar(50), NOT NULL)
List_	The list that this auto-responder belongs to (NULL if belongs to site or server) (varchar(60),

	NULL)
Notify_	Email addresses to notify when this responder is requested (text, NULL)
PgmAfter_	Program to execute after the action of an auto-responder (text, NULL)
PgmBefore_	Program to execute before the action of an auto-responder (Text, NULL)
RespondID_	Unique ID for this responder (int, Identity, NOT NULL)
Site_	What Site does this apply to? This can be the name of a site, the name of a list (in the form of "list \$listname"), or "all" for server-level responders. (varchar(50), NOT NULL)

"phrases_" table

The phrases_ table holds "phrases" words that should trigger a specific action. You define the phrase, where in the message to search, and what to do when the phrase is found.

Primary key:	PhraseID_
AppliesTo_	What fields does this apply to: all, body, subject, from, header, subject-body, etc. (varchar(30), NOT NULL)
Context_	In what context does this apply: <i>\$listname</i> , "all lists", "all sites", or "lyris@". This attribute should work in conjunction with "Site" (below) to locate the phrase at the correct hierarchical level (varchar(20), NOT NULL)
DocTitle_	What document should be mailed/displayed in response to seeing this phrase (varchar(200), NOT NULL)
Notify_	Email addresses to notify when this responder is requested (text, NULL)
PgmAfter_	Program to execute after the action of an action phrase (text, NULL)
PgmBefore_	Program to execute before the action of an action phrase (text, NULL)
Phrase_	The phrase words to search for. If more than one, separated by a carriage return (text, NULL)
PhraseID_	Unique ID for this phrase (int, Identity, NOT NULL)
Replace_	Replace the matched action phrase text with this text (text, NULL)
Site_	Name of site this applies to. This field can be a sitename, a listname (in the form of "list

\$listname"), or "all" to designate a server-level phrase (varchar(50), NOT NULL)

"log_" table

The log_ table keep a record of every email send out by ListManager, except single recipient mail such as subscription confirmations or hello documents. This data is mainly interesting to administrators but is also used by the Lyris ListManager billing program.

Primary key:	LogID_
Body_	The body text of this log entry, if applicable (text, NULL)
ByteCount_	The size of the message, in bytes. (int, NULL)
Created_	Date log entry was created (smalldatetime, NULL)
List _	Name of list this log entry is for, if applicable (varchar(60), NULL)
LogID_	Unique log ID (int, identity, NOT NULL)
RecipientCount_	The number of recipients to receive this message. (int, NULL)
Referer_	Name of the object this log entry is referring to (varchar(50), NULL)
Site_	Name of site this log entry is for, if applicable (varchar(50), NOT NULL)
Source_	Source module that is logging this ('ListManager list posting', 'ListManager autoresponder', 'ListManager digest posting', 'ListManager index posting'). (varchar(50), NOT NULL)
SubsetID_	The ID of the subset used to create the message this log entries details. NULL or 0 if no subset used. (int, NULL)
Title_	English title of this log entry description (refers to documents table) (varchar(200), NULL)
UserName_	Person who caused this log (varchar(100), NULL)

"domainconnectionlimits_" table

This table stores any domains that should have a different number of connections than is the default for the server. This controls the maximum number of connections allowed to a given domain's IP address. Some domains will not allow many connections to their mail servers, and this allows ListManager to be configured to not try and open so many connections. By default ListManager will open up to ten connections per IP address. Opening more connections often causes too much load on the receiving mail server, and may kick off "denial-of-service" attack warnings, which will cause the ListManager server to get blocked/banned. It is STRONGLY suggested that this setting not be increased beyond ten, but in fact be used to limit ListManager for those servers that won't even allow ten connections.

Primary key:	Domain_
Domain_	The domain that should be

	limited in the number of connections allowed. The domain should be in lower-case. (varchar(250), NOT NULL)
ConnectionLimit_	The number of connections to allow, per IP address, to this domain. (smallint, NOT NULL)

"dnsbypass_" table

This table allows users to configure explicit servers to provide DNS information. This is normally used when ListManager is behind a firewall and there are mail servers also behind the firewall that should be connected to directly, without going out the firewall and coming back in. The mail server may not have a publicly known address, so this table provides a location to tell ListManager about an internal DNS server that knows about the internal-only domains.

This table also has the default root servers. If you know that your physical location is best served by a subset of these root servers the additional entries can be removed. These root servers are only used to provide a starting place if no other information is known about the domain. Removing all the root servers will cause ListManager to be unable to send mail. Providing only local DNS servers as the root server may or may not allow ListManager to function, depending on the configuration of that DNS server, but will cause it to perform more slowly. Subsequent requests for data, for example from the GTLD's, will not be asked of the local DNS server, unless these are also bypassed.

Primary key:	UniqueID_
UniqueID_	An auto-numbering field to provide a unique ID for this row. (int, NOT NULL)
Domain_	The domain that this bypass information applies to. The domain should be in lower-case. (varchar(200), NOT NULL)
HostName_	A name for the host. (varchar(200), NOT NULL)
Type_	Is this a MX (15) or an A (2) record type? (tinyint, NOT NULL)
Address_	The address of the dns server, in dot-pair string notation. (ex: 222.11.22.11) (varchar(15), NOT NULL)

"bannedmembers_" table

The list of banned members for this server. Bans can be at a server, site, or list level, and may ban members in a variety of ways - with a complete email address, by banning all members at a given domain, or accepting only named users.

Primary key:	UniqueID_
UniqueID_	Auto-numbering field to uniquely identify this row. (int, NOT NULL)
Domain_	The domain portion of the email address. (varchar (250), NULL)
ListID_	The list that this ban is associated with. NULL for a site or server ban. (int, NULL)
SiteID_	The site that this ban is associated with. NULL for a list or server ban. (int, NULL)

StopLogic_	Logic for processing the ban information. A = always accept this user, C = conditionally accept this user, R = reject this user. (char(1) NOT NULL) UserNameLC_ The user name for this ban, if any, in lower case. (varchar (100) NULL)
------------	--

"config_" table

The config_ table is used to store system-wide configuration values. There are a number of different keys that have meaning to ListManager, and many of these are listed below. In most cases changing values in the config table with direct SQL will not take effect until the system is restarted.

Primary key:	none. (Key_ and MachineName_ should for a unique key)
Key_	The key for finding this data field (varchar(40) NOT NULL)
Value_	The data field value (text NULL)
MachineName_	The machine name that this configuration setting applies to. The machine name comes from the name of the computer. (varchar (25) NOT NULL)

These are the Keys available to put into the Config database:

- AdminEmail
- Autoemail
- BannedFrom
- Bind
- BindSmtpOutgoing
- BindDnsOutgoing
- CheckJoinMailForSpam
- CleanAuto
- DbVersion
- DefaultDomain
- DesiredMaxBandwidth
- DigestFtr
- DigestHdr
- DisableRegexBadFirstLines
- DisableRegexBadFrom
- DisableRegexBadSubject
- ExtDir
- Forward
- GlobalBanServerAddress
- GlobalBanServerPassword
- GlobalBanServerPort
- HeloText
- InmailPurgeDays
- LastMemberIdCheckedForAutoDocSend
- LogFileLines
- LoggingErrorLevel
- LogToConsole
- LogToFile
- LogMax
- LogTime
- LogWhatSubsystems
- LoopBreak
- LoopExcept
- Mailspeed

MaxSubscribeBodyLines
 MaxSubscribeBodyLinesHtml
 MergeAllowBody
 MergeCapabilities
 684
 MessageFtr
 MessageHdr
 MessagesPerHour
 MessSize
 Modnotold
 NntpRecvPort
 NoAddress
 NoExt
 NoFastErrorMail
 NoLog
 NoNNTP
 NoRemPurge
 OutmailPurgeDays
 PgmAfter
 PgmBefore
 RcvDomains
 Rebrandemail
 Rebrandname
 Relay
 RetryTiming
 Rewrite
 Serial
 SmtplRecvPort
 SubsetPermissions
 SystemlpAddresses
 TCLPORTHosts
 TCLPORTPasswd
 TcplPortRecvPort
 TcplPortTimeout
 TcplWebServerAdmin
 ThreadTrack
 ToForward
 ToRewrite
 TranslateLangID
 Url

"sites_" table

The sites_ table stores information about the different sites supported by this ListManager server. Lyris ListManager allows you to define different virtual domains for the outside world. For example, one single ListManager server could support CheapTickets.acme.com, GreatVacations.acme.com, and BanjoDiscussions.banjo.org. To users these look like three different servers, but they are all managed from one ListManager configuration.

Primary Key:	SiteID_
City_	City location of this site (varchar(100), NULL)
Comments_	Additional comments about this site (from document table) (varchar(200), NULL)
Country_	location of this site (varchar(40), NULL)
Desc_	Short description of this site (from document table) (varchar(200), NULL)
DigestFtr_	Text that should be appended to

	every digest on the lists in this site (Text, NULL)
DigestHdr_	Text that should be prepended to every digest on the lists in this site (text, NULL)
DomainName_	Domain name (Hostname) that this site should answer to (varchar(100), NULL)
GuiHost_	The name of the web interface domain used for clickthroughs and other customer interaction (varchar(100), NULL)
IncomingIP_	The IP address to associate with this site for SMTP banner messages (int, NULL)
LicenseLevel_	Limit the site to a lower license level. Useful for hosting situations. Values are (S, P, E) (char(1), NULL)
MailAddr_	Postal mailing address for this site (from document table) (varchar(238), NULL)
MainAdmin_	Reference to main administrator of this site (from people table) (varchar(50), NULL)
MessageFtr_	Text that should be appended to every message on the lists in this site (text, NULL)
MessageHdr_	Text that should be prepended to every message on the lists in this site (Text, NULL)
Name_	Name which identifies this site. e.g.: 'Example Group Ltd' (varchar(30), NOT NULL)
OutgoingIP_	The IP address to use for sending SMTP data for this site. (int, NULL)
OwnerGIF_	URL to GIF for Owner's logo (varchar(100), NULL)
SiteID_	Unique identifier for this site. (int, NOT NULL)
TechSupp_	Reference to technical support contact for this site (from people table) (varchar(50), NULL)
TranslateLangID_	The default translation language for this site. Applies to default list documents if translated versions exist and to the web interface (int, NULL)
URLLyris_	URL to ListManager 5.0 homepage for this site (varchar(100), NULL)
URLOwner_	URL for owner's site. eg: http://www.example.com (varchar(100), NULL)
WWWPort_	What port to use for this web server. eg: 80 (varchar(10), NULL)

"lists_" table

The lists_ table stores configuration information for every list in the system. A list must be associated with a

topic, and the topic with a site. The list settings control many aspects of ListManager processing, from who can post and make changes to which documents are sent for the Hello and Goodbye logic and what headers and footers are inserted into messages.

Primary key:	ListID_
Additional_	Additional fields that should be requested from new subscribers [inactive](text, NULL)
Admin_	Administrator responsible for this list (link to people) (varchar(50), NOT NULL)
AdminSend_	Only Admins can send mail to the list (bool)
AllowCross_	Allows cross-posting of identical messages (bool)
AllowDupe_	Allows posting of identical messages (bool)
AllowInfo_	Allow non-members to read general information about this list (bool)
Anonymous_	If true, the messages posted to the list are stripped of all identifying markers, so that they become anonymous (bool)
AnyonePost_	Anyone can post to the list, even non-members (bool)
ApproveNum_	Number of messages have to be approved before a new person can post unapproved (smallint, NOT NULL)
ArchivDays_	Keep message archives around for how many days (smallint, NOT NULL)
ArchivNum_	Keep how many individual messages archived (int, NOT NULL)
BlkSubjOk_	If set to TRUE, then Lyris will allow messages that have a blank subject line (bool)
Child_	If set, then this list can act as a 'child' list to other lists, meaning that this list can be a member of another list, and receive postings from that list. If this is not set, Lyris automatically rejects posting submissions from other lists as possible error-mail or loop conditions. (bool)
CleanAuto_	Do not mark as 'held' members who bounce too much email (ie: let them bounce all they want) (bool)
CleanDays_	After how many days a user is 'held' should we purge them (smallint, NOT NULL)
CleanNotif_	For how many days after a user is 'held' should we send them reminder messages (smallint, NOT NULL)
CleanUnsub_	After how many days a user is

	'unsub' should we purge them (smallint, NOT NULL)
Comment_	Holds whatever comments the programmer wishes to put in. Useful as a user-defined 'additional info' field. (text, NULL)
CommentsID_	Comment about this list (link to document) (int, NULL)
ConferencePost_	Who is allowed to post to the list's conference: A = admin only, M = members only, E = everyone (char(1), NOT NULL)
ConferenceVisibility_	Who can access the conference: D - disabled conference, M - members only, E - everyone (char(1), NOT NULL)
ConferenceDuration_	how long the contents of a conference are kept in memory: stored in hours. (smallint, NOT NULL)
ConfDays_	How many days after a member has not confirmed to list, should he be deleted (smallint, NOT NULL)
ConfNotify_	At what day interval should held members receive notification (tinyint, NOT NULL)
ConfUnsub_	When unsubscribing over email, should 2=do it immediately, 1=confirm always, 0=confirm only if questionable (tinyint, NOT NULL)
CreatStamp_	Date/time stamp this list was created (smalldatetime, NULL)
CrossClean_	Clean out duplicate recipients with cross-posts, so that members of multiple lists only receive one copy of a message (bool)
DefaultFrom_	The 'from' value that should be used as a default for this list. (varchar(200), NULL)
DefaultTo_	The value for the TO header, if it is not provided (varchar(200), NULL)
DefaultSubject_	The value for the SUBJECT header, if not provided. (varchar(100), NULL)
DescLongID_	Long description of this list (link to document) (int, NULL)
DescShort_	Short description of this list (varchar(200), NOT NULL)
DetectHtmlByDefault_	Should new messages created in the web interface detect HTML by default. (bool)
DetectOpenByDefault_	Should new messages created in the web interface perform open detection by default. (bool)
DigestFtr_	Text that should be appended to every digest (text, NULL)

DigestHdr_	Text that should be prepended to every digest (text, NOT NULL)
Disabled_	If set, then this list is disabled. The list admin is not allowed to send any email to the list (bool)
ErrHold_	After how many days of mail bounces should a member be placed on hold? (default=5) (tinyint, NOT NULL)
ErrReset_	After how many days of not bouncing mail should a member's bounce count be reset to 0 (default=5) (tinyint, NOT NULL)
ExpireDays_	How many days after a member has expired, should he be deleted (smallint, NOT NULL)
From_	If filled in, this specifies the from field of every message send on this list (varchar(100), NULL)
HdrRemove_	Headers that should be removed from list postings (text, NULL)
Hidden_	Controls whether this mailing list is hidden in the web and via email (bool)
KeepOutmailPostings_	How long to keep postings in the outmail_ table. More data gives more accurate reports, but slows the server performance. (smallint, NOT NULL)
Keywords_	Keywords that describe this list (varchar(200), NULL)
ListID_	An incrementing numeric counter for the list (int, NOT NULL)
ListSubj_	Should we prepend the list name to the subject line of every outgoing list message (char(1), NOT NULL)
MaxMembers_	Maximum number of members allowed on this list. If set to 0, then no limit (int, NOT NULL)
MaxMessNum_	Maximum number of messages that can be sent to the list in 24 hour period (0=unlimited) (smallint, NOT NULL)
MaxMessSiz_	In bytes, the maximum size of allowed messages from nonadmin people (int, NOT NULL)
MaxPerUser_	What is the maximum number of messages that a single user can post on a single day to this list (int, NOT NULL)
MaxQuoting_	What is the maximum number of contiguous lines that can be quoted at once in a single message (int, NOT NULL)
MergeAllowBody_	Are Tcl mail merge tags evaluated in the body of the message (tinyint, NOT NULL)

MergeCapabilities_	What capabilities are allowed for Tcl mail merging (tinyint, NOT NULL)
MessRecips_	If set to TRUE, then Lyris will save the member ids of the recipients of each message posting in the messages table (bool)
MessageFtr_	Text that should be appended to every message (text, NULL)
MessageHdr_	Text that should be prepended to every message (text, NULL)
ModHdrDate_	If set, then moderated messages have their Date: header rewritten to be the date/time that they are approved, removing the original Date: header stamp (char(1), NOT NULL)
Moderated_	How often, if at all, is this list moderated? (varchar(20), NOT NULL)
Name_ I	Internal name which identifies this mailing list, not seen by users (varchar(60), primary key, NOT NULL)
NameReqd_	When subscribing over the web, should 0=name is optional, 1=name is not asked for, 2=require name (tinyint, NOT NULL)
NoArchive_	Messages on this mailing list should not be archived (bool)
NoBodyOk_	If set to TRUE, then Lyris will allow messages with a blank body through to the mailing list (bool)
NoConfirm_	Do not send out a confirming message to new subscribers to make sure they used the same email address (bool)
NoEmail_	Reject all submissions by email. Submissions must be by web (bool)
NoEmailSub_	Disallow all subscriptions via email? (bool)
NoListHdr_	If true, the list-help headers are not included (bool)
NoMidRewr_	If set to TRUE, then Lyris will not rewrite the Message-ID, but instead leave it alone, and use X-Lyris-Message-ID (bool)
NoNNTP_	Do not make this group available over NNTP (bool)
NoSearch	Messages on this mailing list should not be full text indexed, and will not be searchable (bool)
PasswdReqd_ W	When subscribing over the web, should 0=password is optional, 1=password is not asked for, 2=require password (tinyint, NOT NULL)

PgmAfter_	Program to execute after a message has been posted to the list (Text, NULL)
PgmBefore_	Program to execute before a message is posted to the list (text, NULL)
PgmSub1_	Program to execute before a member is subscribed (text, NULL)
PgmSub2_	Program to execute after a member is subscribed (text, NULL)
PgmUnsub1_	Program to execute before a member is unsubscribed (text, NULL)
PgmUnsub2_	Program to execute after a member is unsubscribed (text, NULL)
PostPass_	If set, then postings require the user password in the body of the text somewhere. 0=no requirements, 1=required if member has one, 2=all members need a password (tinyint, NOT NULL)
PrivApprov_	Search rules for auto-approving applications to a private list (Text, NULL)
PrivDays_	How many days after a member has applied to a private list, should he be deleted if not approved (smallint, NOT NULL)
PrsrvXTags_	Should we preserve X-.... SMTP header tags (bool)
ReferralPurgeDays_	Purge referral data after how many days? (smallint, NOT NULL)
ReferralsPerDay_	The maximum number of people that a member can refer, per day. (smallint, NOT NULL)
RelHour_	The hour that this message should be automatically approved (tinyint, NOT NULL)
RelPending_	If a pending message hasn't been approved after X days, then send it anyway (tinyint, NOT NULL)
ReplyTo_	If filled in, this specifies the reply-to field of every message send on this list (varchar(100), NULL)
ReviewPerm_	What security permission to put on 'review' command? 0=only admins, 1=only members, 2=anyone (tinyint, NOT NULL)
SMTPFrom_	If filled in, this specifies the SMTP from field of every message send on this list (varchar(100), NULL)
SMTPHdrs_	Text that should be included in the SMTP header of every message (text, NULL)

Security_	Is this list 'private' (admin approval) 'closed' (admin must add you) 'password' (you need the password) 'open' (varchar(8), NOT NULL)
SimpleSub_	Should this list not allow any of the fancy options, such as norepo/noack/password/etc? (bool)
SponsOrgID_	Organization that is sponsoring this list (link to document, type is organization) (int, NULL)
SubNotDays_	What days of the month should a subscriber/unsubscriber report be sent to the list admins? Choices are daily, monthly, monday tuesday wednesday, thursday, friday, saturday, sunday, and modifiers appended to this can be - nochart (do not include chart of activity) and -email (include email addresses) (text, NULL)
SubPasswd_	If this list is 'password' protected for new subscribers, what is the password? (varchar(50), NULL)
To_	If filled in, this specifies the To: field of every message send on this list (varchar(200), NULL)
Topic_	Topic that this list belongs to (varchar(50), NOT NULL)
Transact_	Should Lyris send a transaction notification to list admins during the sending process? This character field controls various possibilities: f = first only, l = last only, b = both first and last, a = all, d = debug all (varchar(5), NULL)
TranslateLangID_	The default translation language for this site. Applies to default list documents if translated versions exist and to the web interface (smallint, NOT NULL)
URLList_	URL to list's homepage (varchar(100), NULL)
URLLogo_	URL to GIF/JPEG list logo (varchar(200), NULL)
VisibGlobl_	Visible via 'list global'? (bool)
Visitors_	Allow non-members to visit the list Web site without joining (they cannot contribute) (bool)

"topics_" table

The topics_ table organizes lists together under a particular site. Multiple lists can be associated with the same topic, and every topic has exactly one site.

Primary key:	Title_
Admin_	Administrator responsible for this list (link to people)

Comment_	(varchar(50), NULL) Comment about this topic (link to document) (varchar(200), NULL)
Desc_	Short description of this topic (link to document) (varchar(200), NULL)
DescLong_	Long description of this topic (link to document) (varchar(200), NULL)
Hidden_	Controls whether this topic is hidden in the web (char(1), NOT NULL)
Keywords_	Keywords that describe this topic (varchar(200), NULL)
SiteName_	What site does this belong to (varchar(30), NOT NULL)
SponsOrg_	Organization that is sponsoring this topic (link to document, type is organization) (varchar(200), NULL)
Title_	Unique title of this topic (varchar(50), NOT NULL)
URLLogo_	URL to GIF/JPEG topic logo (varchar(200), NULL)
URLOrgLogo_	URL to a gif/jpeg logo for this organization (varchar(100), NULL)
URLTopic_	URL to topic's homepage (varchar(100), NULL)

"mimeencodings_" table

This table stores the standard MIME encoding types. Note that the "binary" type is not used in practice and has been omitted.

Primary key:	EncodingID_
EncodingID_	An auto-incrementing ID field that uniquely identifies a record in this table (int, NOT NULL)
Name_	This is the MIME encoding type. It can only be one of 7bit, 8bit, quoted-printable, or base64 (varchar(40), NULL)
Desc_	A text description. Informational only (varchar(120), NULL)

"mimecharsets_" table

This table stores the standard MIME character sets that can be used to create international content.

Primary key:	CharSetID_
CharSetID_	An auto-incrementing ID field that uniquely identifies a record in this table (int, NOT NULL)
Name_	This is the MIME compatible name of the character set (varchar(40), NULL)
Desc_	A text description. Informational only (varchar(120), NULL)

"messagetypes_" table

This table stores information about each internal message that is currently customizable.

Primary key:	MessageTypeID_
MessageTypeID_	An auto-incrementing ID field that uniquely identifies a record in this table (int, NOT NULL)
Name_	The name of this message type. For example "list-hello" (varchar(60), NULL)
Desc_	A text description. Informational only (varchar(120), NULL)
OutmailType_ Type_	This is the value used for the field in the outmail_ table for this message type (varchar(50), NOT NULL)
OutmailTitle_ Title_	This is the value used for the field in the outmail_ table for this message type (varchar(70), NOT NULL)

"docs_" table

The docs_ table stores predefined content that can be used in a variety of situations for sending mail. Documents may be pre-defined responses, such as the Hello or Goodbye document, or it can be content created for mailing to a list.

Primary key:	DocID_
Created_	Date document was created (smalldatetime, NULL)
Desc_	The author's description for this content. (varchar(100), NULL)
DocID_	Unique document ID (int, identity, NOT NULL)
DocType_	Type of the document, reference to doctypes_ table. (varchar(20) NOT NULL)
IsReadOnly_	If true, web interface will not let user change this document. (bool)
IsTemplate_	Is this document a template? (bool)
HdrFrom_	The from header argument that should be sent when delivered via SMTP. (varchar(200) NULL)
HdrTo_	The to header argument that should be sent when delivered via SMTP. (varchar(200) NULL)
NativTitle_	Title of the document, in its native language (varchar(100), NOT NULL)
Site_	Name of site this document belongs to. This field can be a sitename, a listname (in the form of "list:listname"), or "all" to designate a server-level document (varchar(66), NOT NULL)
Title_	Title of the document, in English (varchar(200), NOT NULL)

"doctypes_" table

The doctypes_ table is a lookup table for the allowed document types in the docs_ table. Every DocType_ in the docs_ table should have an entry in the doctypes_ table.

Primary key:	DocType_
Desc_	Textual description of this document type (varchar(100), NOT NULL)
DocType_	Unique ID identifying this document type (varchar(20), NOT NULL)

Document Types supported by ListManager

BILLING	Billing invoice data.
CONTENT	Content created to be sent to a list.
MESSAGE	An internal message sometimes sent to a user or an administrator.

"listdocs_" table

The listdocs_ table maps documents owned by a list to the docs_ table, where the actual content resides.

Primary key:	ListDocID_
ListDocID_	An auto-numbering column to provide a unique identifier for this row. (int, NOT NULL)
IsDefault_	Is this a default document? (bool)
ListID_	The list that this document is associated with. (int NOT NULL)
DocID_	The ID of the document in the docs_ table. (int NOT NULL)
MessageTypeID_	The type of the document, from the messagetypes_ table. (int, NOT NULL)
TranslateLangID_	The language of this document, from the translatelang_ table. (int, NOT NULL)

"sitedocs_" table

This table maps documents owned by a site to the docs_ table, where the actual content resides.

Primary key:	SiteDocID_
SiteDocID_	An auto-numbering column to provide a unique identifier for this row. (int, NOT NULL)
IsDefault_	Is this a default document? (bool)
SiteID_	The site that this document is associated with. (int NOT NULL)
DocID_	The ID of the document in the

MessageTypeID_	docs_ table. (int NOT NULL) The type of the document, from the messagetypes_ table. (int, NOT NULL)
TranslateLangID_	The language of this document, from the translatelang_ table. (int, NOT NULL)

"docparts_" table

The docparts_ table holds the language specific version of content found in the docs_ table. The docs_ table might define a Hello document, for example, which would result in the actual body of the message appearing in the docparts_ table, with one row for each language. Every entry in the docparts_ table must reference a piece of content in the docs_ table.

Primary key:	UniqueID_
Body_	The body portion of the message. (text, NULL)
CharSetID_	The character set from mimecharsets_ (int, NULL)
DocID_	A reference to the content described in the docs_ table. (int, NOT NULL)
EncodingID_	A reference to mimeencodings_ (int, NULL)
HasBeenEncoded_	Has the data in this row already been encoded to this encoding method? (bool)
HdrAdd_	Additional headers that should be added for this message. (Text, NULL)
MimePartName_	The name of the content, for mime purposes. If it is an attachment, for example, is the file name of the attachment. (varchar(255), NULL)
MimeTypeID_ mimetypes_	A reference in to the table telling what mime type this message is. (int, NULL)
UniqueID_	An auto-incrementing field that defines a unique identifier for this row. (int, NOT NULL)

"mimetypetoext_" table

The mimetypetoext_ (Mime Types to Extensions) table is a lookup table for the allowed document types in the docs_ table. Every DocType_ in the docs_ table should have an entry in the doctypes_ table.

Primary key:	none
TypeID_	(int, NOT NULL)
ExtensionID_	(int, NOT NULL)

"mimetypes_" table

The mimetypes_ table is a lookup table defining a numeric identifier for a mime type and providing a description for that type.

Primary key:	TypeID_
TypeID_	Auto-numbering identifier for the

TypeName_	mime type. (int, Primary Key, NOT NULL) The mime type, such as text/html or text/plain. (varchar(40), NOT NULL)
-----------	---

"mimeexts_" table

The mimeexts_ table is a lookup table of a numeric identifier for a mime extension, and the text description of the extension.

Primary key:	ExtensionID_
ExtensionID_	An auto-numbering value to identify the extension. (int, Primary Key, NOT NULL)
ExtensionName_	The name of the mime extension, such as "gif" or "txt". (varchar(10), NOT NULL)

"clickgroups_" table

This table stores the group name for a Group ID identifier.

Primary key:	GroupID_
GroupID_	An auto-numbering field to provide a unique ID. (int, NOT NULL)
GroupName_	A textual description for this group. (varchar(100), NOT NULL)

"clickstreamdata_" table

This table stores additional information about a click through/stream event. The standard information is recorded in the clicktracking_ table, with the additional information provided in this table. Generally only one of these fields is present for each clickstream event, as supported by that technology.

Primary key:	ClickID_
ClickID_	The ClickID_ value from the clicktracking_ table. (int, NOT NULL)
UnitPrice_	The price recorded for this purchase. (numeric (2,2), NULL)
Quantity_	The number of items purchased. (int, NULL)
Points_	The points that this item represents. (int, NULL)
Stage_	The stage that this item represents. (int, NULL)
ProductSku_	The product SKU for this purchase. (varchar(25), NULL)
OrderID_	The Order ID of this purchase. (varchar(25), NULL)
CustomerID_	The customer ID, from the customer's accounting system, for this transaction. (varchar(25), NULL)

"urls_" table

This table stores information about the URL for a message.

Primary key:	UrIID_
UrIID_	An auto-numbering field to server as a unique identifier for the row. (int, NOT NULL)
PrettyName_	The text that the users sees as the link (ex: Click Me). (varchar(100), NULL)
UrlText_	The actual URL (varchar(255), NOT NULL)

"clicktracking_" table

This table stores information about click tracking, click stream, and open events.

Primary key:	ClickID_
ClickID_	An auto-numbering field to provide a unique ID. (int, NOT NULL)
GroupID_	The group that this click is associated with. (int, NOT NULL)
IPAddress_	The IP address of the user clicking on a link. Stored as an int. (int, NOT NULL)
MemberID_	The ID of the member who was sent the email which resulted in this click (int, NOT NULL)
MessageID_	The ID from the outmail_ table that represents the mailing that caused this event. (int, NOT NULL)
StreamWebPageName_	The user-provided name of the web page that caused this event. (varchar(255), NULL)
TimeClicked_	The time that the click event occurred. (smalldatetime, NOT NULL)
UrIID_	The ID from the URLs table that stores this URL. (int, NULL)

"usercolumninfo_" table

This table describes demographics added to the members_ table (or to a new table) so that they can appear as options in the custom charting wizards.

Primary key:	ColumnInfoID_
ColumnInfoID_	An auto-numbering unique identifier for this row. (int, NOT NULL)
ChartType_	The type of chart. (char(1), NOT NULL)
Concept_	What concept should this column fall under, for organizational purposes. (varchar(20), NULL)

Description_	A description of the data stored in this column. (varchar(255), NOT NULL)
FieldName_	The name of the column. (varchar(30), NOT NULL)
TableName_	The table that stores this column (normally members_). (varchar(50), NOT NULL)

"axistypes_" table

This table is a lookup for allowed values in the performancemetrics_ table for the axis of a chart.

Primary key:	AxisType_
AxisType_	A short name for the axis type. (varchar(12) NOT NULL)
TypeDescription_	A description for this axis type. (varchar(255), NULL)

"chartdescription_" table

This table is a lookup for allowed values in the performancemetrics_ table for the axis of a chart.

Primary key:	ChartID_
ChartID_	An auto-numbering field to provide a unique row identifier. (int, NOT NULL)
BackgroundColor_	The RGB value of the background for this chart. (int, NULL)
BucketSize_	The number of entries in each bucket. (varchar (10), NOT NULL)
BucketType_	The type of calculation to apply to the bucket, such as 'sum' or 'std-deviation'. (varchar(20), NOT NULL)
ChartTitle_	A title for the chart that will appear on the PNG image. (varchar(70), NULL)
CreateDate_	The date and time the chart was first created. (smalldatetime, NOT NULL)
Description_	A description of this chart. (varchar(255), NULL)
EndDate_	For time range charts, the date and time that the chart should end with. (smalldatetime, NULL)
GDChartType_	The type of chart (such as pie or line), using the numeric constants from GDChart. (tinyint, NOT NULL)
ModifiedDate_	The date and time the chart was last modified. (smalldatetime, NOT NULL)
Series_	Information about the series in the chart. (text, NULL)
Site_	The site associated with this chart. (varchar(50), NOT NULL)

StartDate_	For time range charts, the date and time that the chart should start with (smalldatetime, NULL)
YAxisLabel_	The textual lable for the Y axis that will appear on the PNG image. (varchar(70), NULL)

"IyrMetricEvents" table

This table stores data about performance for ListManager. Generally every minute aggregate data is saved into this table about events listed in the IyrPerformanceMetrics table. These can then be used to create charts or otherwise analyze performance.

Primary key:	none
MetricKey	The key for this event, from the IyrPerformanceMetrics table. (int, NOT NULL) Value The actual data value. (int, NOT NULL)
EventTime	The time and date of this event (datetime, NOT NULL)

"IyrPerformanceMetrics" table

This table stores data about performance metrics used by ListManager. Generally every minute aggregate data is saved in the IyrMetricEvents table for events listed in this table. These can then be used to create charts or otherwise analyze performance.

Primary key:	MetricKey
MetricKey	An auto-numbering value to serve as a unique key for this table. (int, NOT NULL)
Description	A textual description of this metric. (varchar(255), NOT NULL)
ShortName	A short textual unique name for this metric. (varchar(12), NOT NULL)

"IyrLookupCompletionStatus" table

This is a lookup table for the possible completion statuses.

Primary key:	CompletionStatusID
CompletionStatusID	The unique key for this row (int, NOT NULL)
Description	The description of this completion status. (varchar(255), NOT NULL)

"IyrCompletedRecips" table

The IyrActiveRecips table holds recipient rows for mail sending. Each recipient has a row in this table while mail is being delivered to them. Once no more attempts will be made to deliver this recipient the mail message, the row is moved from this table in IyrCompletedRecips.

Primary key:	RecipientID
RecipientID	A unique, autoincrementing identifier for this row (big int, NOT NULL)
BounceMailingID	The identifier for the row in inmail_ that holds the bounce message for this recipient. (int, NULL)
CompletionStatusID	The recipients final completion status, from the lyrLookupCompletionStatus table. (int, NOT NULL)
Domain	The domain part of the recipient's email address. (varchar(250), NOT NULL)
FinalAttempt	The time and date of the last attempt to deliver mail for this recipient. May be the same as FirstAttempt if only one attempt was made. (smalldatetime, NULL)
FirstAttempt	The date and time that the first attempt was made to deliver to this recipient. (smalldatetime, NULL)
MailingID	The mailing that this recipient should receive, from the outmail_ table. (int, NOT NULL)
MemberID	If this recipient is a member their identifier from the members_ table. (int, NULL)
SendTry	The number of attempt that have been made to deliver mail for this recipient. (tinyint, NOT NULL)
TransactionLog	The textual details of the attempt to send to this recipient. Populated based on list settings. (text, NULL)
UserName	The user name portion of the recipient's email address. (varchar(100), NOT NULL)

"inmail_" table

The inmail_ table hold messages received by ListManager. Every received message should be a row in this

table, except for some bounce messages, if these are configured to not be saved. The data is saved as soon as it is received, and then is processed as resources are available. Once the message is processed its status is changed to 'done'.

Primary key:	MessageID_
BeingProcessedBy_	The machine name that is currently processing this inmail record. For use in clustering. (varchar(25), NULL)
Body_	The body of the message (text, NULL)
ByEmail_	Was this current entry created by email, or by some other means, such as a script or web interface (bool)
Created_	Date the message was entered in the queue (datetime, NULL)
DeliverBefore_	After this time the message should not be delivered. (smalldatetime, NULL)
HdrAll_	The complete message header (text, NULL)
HdrDate_	The 'date:' header (varchar(100), NULL)
HdrFrom_	The 'from:' header (varchar(238), NULL)
HdrFromSpc_	The 'from' header (varchar(200), NULL)
HdrSubject_	The 'subject:' header (varchar(200), NULL)
HdrTo_	The 'to:' header (varchar(200), NULL)
List_	The list that this message belongs to (varchar(60), NULL)
MaxRecips_	For a list posting, the maximum number of recipients allowed to get this message. ListManager will remove any members so that this max is not reached. (int, NOT NULL)
MessageID_	Unique message ID (int, identity, NOT NULL)
ModerApp_	Has this message been moderation approved? (bool)
Priority_	The relative importance of processing this incoming message (tinyint, NOT NULL)
PurgeMess_	For a list posting, indicates that the posting should remove recipients who have received the following archived message ids. (text, NULL)
Status_	Status of the message: processed (varchar(20), NOT NULL)
SubsetID_	The ID of the subset sent to for this inmail. NULL or 0 if no subset was used. (int, NULL)
Title_	The title of this message (varchar(70), NULL)
To_	The email address this message

	was sent to (varchar(200), NULL)
Transact_	Transaction log for each record (text, NULL)
Type_	Message type: list, command, error-mail, auto-refused, bad-toaddress (varchar(10), NULL)

"lyrActiveRecips" table

The lyrActiveRecips table holds recipient rows for mail sending. Each recipient has a row in this table while mail is being delivered to them. Once no more attempts will be made to deliver this recipient the mail message, the row is moved from this table in lyrCompletedRecips.

Primary key:	RecipientID
RecipientID	A unique, autoincrementing identifier for this row (big int, NOT NULL)
Domain	The domain part of the recipient's email address. (varchar(250), NOT NULL)
FirstAttempt	The date and time that the first attempt was made to deliver to this recipient. (smalldatetime, NULL)
FullName	The recipient's full name, if known. Copied from the members table for efficiency in mail merging. (varchar(100), NULL)
MailingID	The mailing that this recipient should receive, from the outmail_ table. (int, NOT NULL)
MemberID	If this recipient is a member their identifier from the members_ table. (int, NULL)
NextAttempt	When the next attempt should be made to deliver to this recipient. (smalldatetime, NOT NULL)
NodeID	The identifier of the node that is currently sending this recipient. For clustering purposes. (tinyint, NULL)
NodeSequence	An incrementing value used to ensure locking between nodes works properly. (int, NOT NULL)
SendTry	The number of attempt that have been made to deliver mail for this recipient.

TransactionLog	(tinyint, NOT NULL) The textual details of the attempt to send to this recipient. Populated based on list settings. (text, NULL)
UserName	The user name portion of the recipient's email address. (varchar(100), NOT NULL)

"moderate_" table

The moderate_ table holds messages that need to be moderated, or are set to be released at a particular time. When a message is received it will start with an inmail_ record. If the message needs approval, the record will be copied to the moderate_ table. When it is approved (if it is not also rescheduled) it is copied again to the inmail_ table as an approved message.

Primary key:	MessageID_
AutoApDate_	Date/time at which time this message is automatically approved (smalldatetime, NULL)
Body_	The body of the message (text, NULL)
ByEmail_	Was this current entry created by email, or by some other means, such as a script or web interface (bool)
DeliverBefore_	After this time the message should not be delivered. (smalldatetime, NULL)
HdrAll_	The complete message header (text, NULL)
HdrDate_	The 'date:' header (varchar(100), NULL)
HdrFrom_	The 'from:' header (varchar(200), NULL)
HdrFromSpc_	The 'from ' header (varchar(200), NULL)
HdrSubject_	The 'subject:' header (varchar(200), NULL)
HdrTo_	The 'to:' header (varchar(200), NULL)
List_ T	he list that this message belongs to (varchar(60), NOT NULL)
MaxRecips_	For a list posting, the maximum number of recipients allowed to get this message. ListManager will remove any members over this limit. If this field is not set, then there is no max. (int, NOT NULL)
MemberID_	MemberID of the user who contributed this moderated message (int, NOT NULL)
MessageID_	Unique message ID (int, Identity, NOT NULL)
ModHdrDate_	If set, this moderated message should have its Date: header

Date:	rewritten to be the date/time it is approved, removing the original header stamp (char(1), NOT NULL)
PurgeMess_	For a list posting, indicates that the posting should remove recipients who have received the following archived Message IDs. A space-separated list of IDs. (text, NULL)
ReSubmit_	If set, means that when the message is approved, it should be resubmitted for automatic approval after so many hours (smallint, NOT NULL)
Status_	Status of the message: new, processed (varchar(20), NOT NULL)
SubsetID_	The ID of the subset used to create this message. NULL or 0 if no subset was used. (int, NULL)
Title_	Optional title given to this posting -- can only be set independently of the message (ie with web posting or with 'moderate' commands (varchar(70), NULL)
To_	The email address this message was sent to (varchar(100), NULL)
Transact_	The log of activity for this message. (text, NULL)
Type_	Message type: unknown, command, error-mail, autorefused, bad-to-address (varchar(20), NULL)

"outmail_" table

The outmail_ table stores mail jobs that have been approved to be sent. Records that do not have any recipient left to deliver are automatically purged from outmail_ based on the list setting for "Keep mailing and clickthrough data for how many days". The outmail_ table stores the message text and information needed to sent the message. The recipients who will receive this mail are stored initially in lyrActiveRecips, and will be moved to lyrCompletedRecips as they are completed. With subset postings it is possible to have recipients from multiple lists, but only one list is used for the purpose of mail merging. That is the value in the List_ column.

Primary key:	MessageID_
Body_	The body of the message (text, NULL)
Checksum1_	A value to represent the content of the message body (int, NULL)
Checksum2_	A value to represent the content of the message body (int, NULL)
Checksum3_	A value to represent the content of the message body (int, NULL)
Checksum4_	A value to represent the content of the message body (int, NULL)
Created_	Date the message was entered in the queue (datetime, NOT

	NULL)
DeliverBefore_	After this time the message should not be delivered. (smalldatetime, NULL)
From_	Who the email message is to be from (an email address) (varchar(200), NULL)
HdrAll_	The complete message header (Text, NULL)
InmailBodySize_	The size of the message body from inmail_ (before modifications were made to outmail_) (int, NULL)
InmailHdrFrom_	The HdrFrom value from inmail_. Used to detect crosspostings and duplicate postings. (varchar(238), NULL)
InmailID_	The ID of the inmail record that created this entry. NULL or 0 if no inmail record exists. (int, NULL)
List_	The list that this message belongs to (varchar(60), NULL)
MessageID_	Unique message ID (int, Identity, NOT NULL)
Priority_	The relative importance of processing this outgoing message (tinyint, NOT NULL)
Title_	Title created by the author for this message. (varchar(70), NULL)
To_	Who the message is destined to (an email address) (varchar(200), NULL)
Transact_	Transaction log for each record (text, NULL)
Type_	Message type: list, error-mail, owner-moderated-posting, owner-mail, debug, index, digest, mdigest, unknown (varchar(50), NOT NULL)
SubsetID_	The ID of the subset used to create this entry. NULL or 0 if no subset was used. (int, NULL)

"members_" table

The members_ table stores information about every list member on the server. Members are organized by list, and the same member may be on multiple lists. In this case there will be multiple rows in the members_ table, one for each list that the member has subscribed to.

Internet standards are that the email address should preserve the case of the name, but not make a distinction for comparisons. To support this, and to make searching for members fast, the email address is represented twice in the table: once in full casesensitive form (the EmailAddr_ column), and broken into case-insensitive pieces into columns UserNameLC_ and Domain_. ListManager will automatically populate these columns when a new member is added, but if you want to add members to this table directly using SQL you MUST populate all the fields with the correct values. UserNameLC_ and Domain_ should hold the lower-case version of the portions of the email address, while the full case-sensitive version is stored in EmailAddr_. UserNameLC_ should hold the "user name" portion of the email address, i.e., everything to the left of the @ sign. Everything to the right of the @ sign should be stored in the Domain_ column. Neither the UserNameLC_ or the Domain_ should contain an @.

Information about the state of the user (such as on-hold or unsubscribed) is stored in the members_ table, as well as information about recent bounces. This information allows ListManager to automatically handle administrative tasks such as putting user on hold, taking them off hold, purging users, requiring approvals before joining, etc. Many of the exact options for these are set in the lists_ table.

The members_ table has two fields that are not populated by ListManager, and are available for you to put any data desired. These are UserID_ for up to 20 characters of text, and Additional_ for unlimited text. Additional columns may be added to the table as long as the columns allow NULL or have default values. Adding columns that are NOT NULL will stop ListManager from being able to add new members, so is not recommended. Additional fields in the members table is an easy way to add information that is available for mail merging. However, it is also easy to create an SQL join to other tables, so that is generally a more flexible and efficient way to make your data available for mail merging. You should be careful about adding too many columns to the members table because as the table grows "wider" it will become slower and slower to get information from it, which will impact ListManager's ability to send mail quickly.

Primary key:	MemberID_
Unique index:	Domain_, UserNameLC_, List_
Additional_	Place holder for any information you want to store associated with this member. (text, NULL)
AppNeeded_	Can this person skip bypass approval to send messages? (bool)
CanAppPend_	As a list admin, can this member approve pending (moderated) messages? (bool)
CleanAuto_	Do not mark as 'held' this members if they bounce too much email (ie: let them bounce all they want) (bool)
Comment_	Holds whatever comments the programmer wishes to put in. Useful as a user-defined 'additional info' field (Text, NULL)
ConfirmDat_	Date the user was last sent a 'confirm' message (smalldatetime, NULL)
DateBounce_	The date of the most recent bounce (smalldatetime, NULL). As of version 7.8, this column is no longer used.
DateHeld_	Date the user was held (smalldatetime, NULL)
DateJoined_	Date when person became a member of this list (smalldatetime, NULL)
DateUnsub_	Date when person unsubscribed from this list (smalldatetime, NULL)
Domain_	Lower case version of the domain portion of email address (microsoft.com, for example). (varchar(250), NOT NULL)
EmailAddr_	Full case-sensitive Internet email address (varchar(100), NOT NULL)
ExpireDate_	Membership expires on this date (smalldatetime, NULL)
FullName_	Full name of this person (varchar(100), NULL)

IsListAdm_	Whether this person a list admin of this list (bool)
List_	What list is this person a member of? (link to lists) (varchar(60), NOT NULL)
MailFormat_	What format does the user want to receive mail, (T)ext, (M)ultipart, or (H)TML. (char(1) NOT NULL)
MemberID_	Unique member ID (int, identity, NOT NULL)
MemberType_	What kind of member is this? ('normal', 'confirm', 'private', 'expired', 'held', 'unsub') (varchar(20), NOT NULL)
NoRepro_	Member should not receive a copy of their own posting (bool)
NotifyErr_	For list admins: receive error mail? (bool)
NotifySubm_	List admin: receive notification of pending moderated messages (bool)
NumAppNeed_	How many more approvals does this person need before they can send messages unapproved? (smallint, NOT NULL)
NumBounces_	The number of recent bounces this person has produced (smallint, NOT NULL). As of version 7.8, this column is no longer used.
Password_	Password this person uses for restricted functions (varchar(50), NULL)
RcvAdmMail_	For list admins: receives email messages destined for list admins (bool)
ReadsHtml_	Does the mail recipient read HTML mail? (bool)
ReceiveAck_	Receive an acknowledgement when contributing a posting? (bool)
SubType_	Kind of subscription (digest, index, nomail, mail) (varchar(20), NOT NULL)
UserID_	Holds the user-definable 'user id' information, such as a key back to another table. (varchar(20), NULL)
UserNameLC_	The user name (portion before the @) from the EmailAddr_ column, in lower case. (varchar(100), NOT NULL)

"listmessagesmapping_" table

This table provides NNTP with per-list unique incrementing message IDs. The messages_ table has an auto-numbering field which is incremented for every new archive, regardless of which list it came from. Having large gaps in message IDs causes problems for many NNTP viewers, so this table provides a mechanism to make per-list IDs that will not have gaps between the messages for that list, unless messages are deleted.

Primary key:	none
ListID_	Reference to the ListID_ in table lists_ (int, NOT NULL)
ListMessageID_	A unique key for use by NNTP. (int, NOT NULL)
MessageID_	Reference to the MessageID_ in table messages_. (int, NULL)

"messages_" table

The messages_ table stores the archive of all message postings. This data is required for any searching ability of messages.

Primary key:	MessageID_
Body_	The body of the message (text, NULL)
CreatStamp_	Date/time stamp this message was created (smalldatetime, NOT NULL)
Digested_	Has this message been sent in a digest yet? (char(1), NOT NULL)
HdrAll_	The complete message header (text, NULL)
HdrDate_	The 'date:' header (varchar(100), NULL)
HdrFrom_	The 'from:' header (varchar(200), NULL)
HdrFromSpc_	The 'from ' header (varchar(200), NULL)
HdrSubject_	The 'subject:' header (varchar(200), NULL)
HdrTo_	The 'to:' header (varchar(200), NULL)
List_	The list that this message belongs to (varchar(60), NOT NULL)
MemberID_	Unique member ID of the member who posted this message (int, NOT NULL)
MessageID_	Unique message ID (int, primary key, identity, NOT NULL)
ParentID_	The message thread that this message is responding to, or the ID of this message, if this is the start of a thread. (int, NULL)
SubsetID_	The ID of the subset used to create this message. NULL or 0 if no subset used. (int, NULL)
Title_	Optional title given to

this posting -- can only be set independently of the message (ie with web posting or with 'moderate' commands (varchar(70), NULL)

"lyrPermissionGroups" table

The lyrPermissionGroups table holds the name and description of permission groups for web page permissions.

Primary key:	GroupID
	GroupID The primary key for this group (int, NOT NULL)
GroupName	Name of group, (varchar (50), NOT NULL)
GroupDescription	Description of group, (varchar (200) NOT NULL)

"lyrPermissionGuiAttribs" table

The lyrPermissionGuiAttribs table holds the rules for each group in web page permissions.

Primary key:	AttributeID
AttributeID	Primary key for this rule (int, NOT NULL)
PermissionGroupID	GroupID this rule belongs to (int, NOT NULL). Reference to GroupID on lyrPermissionGroups.
SortOrder	Order in which this rule is applied (int. NULL)
GuiPath	String to match against URI. Determines whether this rule matches varchar (255).
Permission	What to do if this rule matches A=allow, D=deny. Varchar (1) must be A or D.
AttributeDescription	Description of this rule, varchar (100).

"subsets_" table

The subsets table stores information about every subset created in the ListManager system. Subsets allow the list administrator to provide the SQL statement that is used to generate the member list for a posting. The SQL statement is broken into several parts in this table to allow it to be recombined in a generic way across multiple database vendors.

Subsets must be attached to a list, but a list may have multiple subsets.

Primary key:	SubsetID_
ClauseAdd_	SQL statements that would be appended to the end of the SQL statement, such as 'group by' or 'having'. Currently unimplement. (text, NULL)
ClauseAfterSelect_	Data to be added to the subset

	SQL after the select clause. Designed to allow Oracle user to provide a index hint. (text, NULL)
ClauseFrom_	The SQL 'from' line. This is the list of all the tables participating in the SQL statement. (text, NULL)
ClauseOrderBy_	The 'order by' SQL statement. This is for display purposes only. (text, NULL)
ClauseSelect_	The 'select' line of the SQL statement. These are the columns to retrieve. (text, NULL)
ClauseWhere_	The 'where' portion of the SQL statement. Defines the reductions and join rules used to retrieve data. (text, NULL)
Desc_	A short description of the subset. (varchar(60), NULL)
List_	The list that this subset is associated with. (varchar(60), NOT NULL)
Name_	The name of the subset. This is used as name-list@server for email postings. (varchar (60), NOT NULL)
NumTestRecords_	The number of records to retrieve when testing the subset. (int, NOT NULL)
AddWhereList_	Should ListManager automatically add the 'where' criteria to restrict this query to the list name that this subset is associated with, or can this subset get to data outside of this list. Default is T, meaning to add the where criteria and keep the subset restricted to the specific list. Setting this to F allows postings to every member on the server, so is very dangerous. (bool)
AddWhereMemberType_	Should ListManager automatically add the 'where' criteria to restrict this query to 'normal' members? Default is 'T'. Normally ListManager would only send to 'normal' members, not to held or unsubscribed or other statuses. Setting this field to 'F' sends to every member type in the system, so is very dangerous. (bool)
AddWhereSubType_	Should ListManager automatically add the 'where' criteria to restrict this query to 'mail' members? Default is 'T'. Normal postings only go to members who want normal 'mail' delivery, not those wanting

digest or who have requested not to be sent mail. Setting this field to 'F' allows every member type to be sent to, so is very dangerous. (bool)

SubsetID_ Identifier for the subset. (int, identity, NOT NULL)

"translatekeys_" table

This table stores the data that represents the strings that are translatable. Strings longer than 255 characters are specified using a tag rather than the string itself.

Primary key: TranslateKeyID_

TranslateKeyID_ An auto-incrementing ID field that uniquely identifies a record in this table. (int, identity, NOT NULL)

Key_ The string used as the key to the translated value. Will generally be the actual string being translated but could also be a tag. (varchar(255), NULL)

Type_ The type of this string. Only keys of type "text" are actually translated. The rest refer to images. (varchar(40), NULL)

ImgPath_ For image types, this is the relative path to the image. (varchar(60), NULL)

"translatelang_" table

This table stores information about each translated language.

Primary key: TranslateLangID_

TranslateLangID_ An auto-incrementing ID field that uniquely identifies a record in this table (int, identity, NOT NULL)

Name This is the language name in English (varchar(60), NOT NULL)

Path_ This is the relative path to the translated images directory (varchar(60), NOT NULL)

Desc_ A text description. Informational only (varchar(120), NULL)

CharSetID_ A reference to mimecharsets_, this defines the charset used by this language (int, NULL)

"translatevalues_" table

This table stores the translated values for each key and language.

Primary key: TranslateValueID_

TranslateValueID_ An auto-incrementing ID field that uniquely identifies a record

TranslateLangID_	in this table (int, NOT NULL) A reference to translatelang_, this defines what language this translated value is in (int, NOT NULL)
TranslateKeyID_	A reference to translatekeys_, this defines what key is used to access this translation (int, NOT NULL)
IsValid_	When a translatekeys_ record is modified, any translatevalues_ records that reference it are flagged as invalid so that they can be retranslated (bool)
Value_	This is the translated value for the specified language and key (text, NULL)

"referrals_" table

This table keeps information about referrals, aka, Send a Friend or Referrals.

Primary key:	none
SenderMemberID_	The member ID of the originator of the referral. (int, NOT NULL)
SenderEmailAddr_	The email address the sender provided when initiating the referral, which could be different than the email address in the member record. (varchar(100), NULL)
SenderIPAddress_	The IP address recorded from the sender at the time of referral. (int, NOT NULL)
ReceiverMemberID_	The member ID of the 'friend' who should receive this referral. (int, NOT NULL)
ContentID_	The ID of the content to send as the referral. (int, NOT NULL)
RollupID_	(int, NOT NULL)
TimeSent_	The date and time the referral was sent. (smalldatetime, NULL)

"messengerelationship_" table

This table tracks referral outmail objects to the message that first caused the referral action.

Primary key:	none
ParentID_	(int, NOT NULL)
ChildID_	(int, NOT NULL)

"IyrTaskScheduler" table

This table stores the current jobs scheduled to be run.

Primary key:	ScheduleID
ScheduleID	A unique, autoincrementing Identifier for this row. (int, NOT NULL)

RunTime	When should this task be run? (datetime, NOT NULL) LastRunTime When was this task last run? (datetime, NULL)
TaskID	What task should be run, from the lyrTaskDescription table. (int, NOT NULL)
TaskData	Any integer data needed by the task, such as a mailing ID. (int, NULL)

"lyrTaskDescription" table

This table stores the possible tasks that can be run on a scheduled basis by a node in the cluster.

Primary key:	TaskID
TaskID	The unique identifier for this task. (int, NOT NULL)
Description	The textual description for this task. (varchar(255), NOT NULL)
RescheduleMinutes	The number of minutes to wait before running this task again. Zero minutes is a one-time task. (int, NOT NULL)

"lyrSurveyResponseAnswer" table

This table stores the responses to a specific question in a survey. Every survey response will have one row in the lyrSurveyResponse table, and potentially multiple rows in this table, one for each question in the survey.

Primary key:	ResponseID, QuestionID, AnswerID
ResponseID	The identifier of the response from the lyrSurveyResponse table. (int, NOT NULL)
AnswerID	The identifier of the answer from the lyrSurveyAnswers table. (int, NOT NULL)
QuestionID	The identifier of the question from the lyrSurveyQuestions table. (int, NOT NULL)
FreeFormAnswer	Any free form response to a question. (text, NULL)

"lyrSurveyResponse" table

This table stores the responses to a survey. Every survey response will have one row in this table, and potentially multiple rows in the lyrSurveyResponseAnswer table, one for each question in the survey.

Primary key:	ResponseID
ResponseID	An auto-incrementing unique identifier (int, NOT NULL)
MailingID	If this survey was sent via email, what was the mailing ID, from the outmail_ table? (int, NULL)
MemberID	If this response was from a member, what is their member ID, from the members_ table? (int, NULL)
ResponseTime	When was this response received? (smalldatetime, NOT NULL)
WebDocID	The ID from the lyrWebDocs table that represents this survey in HTML form. (int, NOT NULL)
Weight	The relative weight of this response. (int, NULL)

"lyrSurveyQuestions" table

This table describes a question on a survey.

Primary key:	QuestionID
QuestionID	An auto-incrementing unique identifier (int, NOT NULL)
UserQuestionNumber	What question number should this be for the user? (int, NOT NULL)
UserPartNumber	If this is a question that has multiple answers, such as a matrix, what part number is this of the question? (smallint, NOT NULL)
QuestionText	The text of the question. (varchar(255), NOT NULL)

"lyrSurveyAnswers" table

This table describes a possible answer to a question, such as "Strongly agree".

Primary key:	AnswerID
AnswerID	An auto-incrementing unique identifier (int, NOT NULL)
AnswerText	The text of this answer. (varchar(255), NOT NULL)
AnswerCollectionID	If this answer is part of a collection, such as a matrix, what is the collection ID from the lyrSurveyAnswerCollections . (int, NULL)
AnswerOrder	The order that the answers should be provided to the user. (smallint, NOT NULL)
AnswerWeight	The weight that this answer should be considered. Some questions have much more importance, so this allows the answer to be considered more or less important than other answers. (int, NULL)

"lyrSurveyAnswerCollections" table

This table associates an integer identifier with a group of answers to a single survey question. This is used for questions that allow multiple answers, such as a matrix.

Primary key:	CollectionID
CollectionID	An auto-incrementing unique identifier (int, NOT NULL)
CollectionShortName	A short name to refer to this row, as an alternative key. (varchar(15), NOT NULL)
CollectionDescription	A textual description of this collection. (varchar(255), NULL)

"lyrWebDocs" table

The lyrWebDocs table holds HTML pages that are returned to a user's web browser. These documents are currently used only for Member Profile and Referral forms.

Primary key:	WebDocID
WebDocID	A unique ID for this record (int, identity, NOT NULL)
Title	The title of this HTML document. This is only used by the

	administrator's web interface for identification and is not visible to the user. (varchar(60), NOT NULL)
Descript	A description for this HTML document. This is only informational and is only visible to administrators in the web interface. (varchar(120), NULL)
Body	This is the content of the HTML page. (text, NULL)
WebDocTypeID	A foreign key into the lyrLookupWebDocTypes table. This indicates what type of HTML document this is. (int, NOT NULL)
ListID	A foreign key into the lists_ table. This indicates what list this document is associated with. If this is NULL and SiteID is NULL then this is a server HTML document. (int, NULL)
SiteID	A foreign key into the sites_ table. This indicates what site this document is associated with. If this is NULL and ListID is NULL then this is a server HTML document. (int, NULL)
IsReadOnly	Indicates whether this is a document that is populated during database creation or upgraded and can be overwritten during future upgrades. (bool)
IsTemplate	Indicates whether this is a template. (bool)

"lyrLookupWebDocTypes" table

The lyrLookupWebDocTypes table holds type information that is referenced by the lyrWebDocs table.

Primary key:	WebDocTypeID
WebDocTypeID	A unique ID for this record (int, identity, NOT NULL)
Title	The title of this document type. This is only used by the

Descript	administrator's web interface for identification and is not visible to the user. (varchar(60), NOT NULL) A description for this document type. This is only informational and is only visible to administrators in the web interface. (varchar(120), NULL)
----------	---