Dynamic Query Documentation

19 June 2018

Overview

LISTSERV version 15.5 can execute on-demand *Dynamic Queries* against either LDAP or traditional DBMS servers, and use the results for access control and mail delivery. For instance, an LDAP query against an Active Directory server could be used to grant list owner privileges to all members of a particular Windows Security Group. A DBMS query could be used to combine employee rosters for two departments and send a single copy of an announcement to each unique employee e-mail address. Support for other types of data sources can also be provided by writing an exit (short script) to query the custom data source and return the results to LISTSERV.

This document will not cover basic DBMS or LDAP configuration. Please refer to the *"LISTSERV LDAP Documentation"* for information on how to configure LDAP servers in LISTSERV.

Dynamic Query concepts

A *Dynamic Query* is a pre-configured search against either an LDAP directory or a DBMS that returns, at a minimum, a series of e-mail addresses, and may also return additional information, such as the recipient's full name or phone number. Queries can be used either for access control purposes (for instance, to grant all members of the CORP_HR Windows Security Group permission to post to the HR-NEWS list), or to provide additional recipients for a posting to a mailing list.

All dynamic queries are defined ahead of time by the LISTSERV administrator. List owners cannot create their own queries, but they can make use of pre-defined queries and supply parameters to these queries (see below for more information on security restrictions). The LISTSERV administrator determines how these parameters are used. For instance, the administrator could define a query that matches all members of a particular Windows Security Group, specified as a parameter. List owners could then provide the group name when they refer to the query. Queries whose definition contains no parameters cannot be altered by list owners.

Dynamic queries are executed when, and only when, LISTSERV needs access to the query results to complete a task. A dormant query consumes no resources. Note that LISTSERV does execute queries when verifying the syntax of list header changes. LISTSERV rejects any invalid queries and issues a warning when the query is valid, but returns no data, as this usually means that a parameter was spelled incorrectly.

Queries can be inserted in most list header keywords used for access control or to enumerate e-mail addresses. Example:

** Owner= J.Smith@example.com (Joe Smith)*
** Owner= Query(DEPT,HR) (HR Department)*
** Review= Private,Query(DEPT,HR),*@example.com*

Queries can also be used to create dynamic sub-lists, which are covered in a separate chapter.

Defining Dynamic Queries

Each dynamic query must be given a nickname, which may not contain white-space or non-printable characters. List owners know the queries by their nicknames, and have no access to their actual definitions. A query is defined by a LISTSERV configuration variable called:

DYN_QUERY_*nickname*=kwd1 [kwd2 […]]

On unix, the nickname must be entered in upper case, and shell escape and quoting conventions must be respected. Keywords have the following syntax:

keyword=token
keyword='string that can contain white space'
keyword="string that can contain white space"

Quotes within a quoted string must be doubled. The following keywords are recognized for both LDAP and DBMS queries:

- **TYPE=**LDAP | DBMS | EXIT (default: LDAP)
  Whether the query uses LDAP, DBMS or a customer-supplied exit (script).
- **SERVER=***server_name* (default: DEFAULT)
  The nickname of the LDAP or DBMS server against which to run the search. If omitted, the search is run against the default (unnamed) LDAP or DBMS server.
- **E-MAIL=***attr_name* (default: see explanation)
  EMAIL=*attr_name* (default: see explanation)
  NAME=*attr_name* (default: see explanation)
  The name of the LDAP attribute or SQL column that contains the e-mail address or full name of the recipient, respectively. For LDAP, a default value can be specified in the LDAP server configuration (LDAP_DEFAULT_EMAIL_*server* and LDAP_DEFAULT_NAME_*server*). There is no default value for DBMS sources. The e-mail attribute or column name must be supplied, either explicitly or through LDAP_DEFAULT_EMAIL; the name is optional.

The following keywords are recognized for LDAP queries only:

- **SCOPE=***scope* (default: SUBTREE)
  The scope of the LDAP query.

- **BASE=***DN*
  The 'distinguished name' of the search base. Mandatory; **may refer to parameters.**
- **FILTER=***filter*
  The search 'filter'. Mandatory; **may refer to parameters.**
- **ATTRS=***attr1 [attr2 […]]*
  Any additional attributes (besides the name and e-mail address) that should be returned by the query when it is used in the context of processing a posting to a mailing list. These additional attributes become available as mail-merge fields. This keyword is ignored when the query is executed for other purposes (access control, etc.)

These keywords are recognized for DBMS queries only:

- **DBMS=**ODBC | OCI | CLI | UODBC (default available if only one driver is configured)
  The name of the DBMS driver to use for the search. As with other DBMS functions, a default is provided if only one DBMS driver has been configured. Otherwise, this attribute is mandatory.
- **SEARCH=***SELECT_statement*
  The SELECT statement to execute. It must return at least the e-mail column, and must also return the name column if one was specified. It may return additional columns, which are made available as mail-merge fields when the query is used in the context of processing a posting to a mailing list. For optimal performance, you should list the e-mail and name columns first. This attribute is mandatory and **may refer to parameters.**

These keywords are recognized for TYPE=EXIT queries only:

- **EXITPARMn=***arbitrary_data* ($1 \le n \le 4$)
  Four independent keywords are provided to pass arbitrary data or arguments to the exit. These keywords are made available to the exit script as it is called and **may refer to parameters.**

For security reasons, keywords may not refer to parameters unless indicated otherwise. **LISTSERV inserts all parameters as escaped text constants,** to prevent them from changing the semantics of the search.

Parameters are specified as %1 for the first parameter, %2 for the second one, and so forth. The percent sign must be doubled.

Query examples

These examples have been wrapped for legibility, but they must be entered as a single line in the LISTSERV configuration. Parameters are shown in red for emphasis only.

- Query matching Active Directory users in a particular organizational unit at EXAMPLE.COM:

  *SERVER=*nickname *BASE=OU=%1,DC=EXAMPLE,DC=COM*
  FILTER= '(&(objectcategory=person)(objectclass=user))'

- Query matching members of a Windows Security Group at EXAMPLE.COM:

  *SERVER=*nickname *BASE=CN=Users,DC=EXAMPLE,DC=COM*
  FILTER= '(&(objectcategory=person)(objectclass=user)
  (memberof=CN=%1,CN=Users,DC=EXAMPLE,DC=COM))'

- Advanced example: query matching recipients from a DBMS table whose e-mail address is at a particular hostname.

  *TYPE=DBMS SERVER=*nickname *SEARCH='SELECT CUST_EMAIL,CUST_NAME FROM SOMETABLE WHERE CUST_EMAIL LIKE CONCAT(''%%@'',%1)' E-MAIL=CUST_EMAIL*

  Because LISTSERV inserts all parameters as escaped string constants, it is not possible to form the pattern '%@*[parameter #1]*' by entering '%%@%1' in the SELECT statement. Instead, the run-time CONCAT function is used. The quotes surrounding the first parameter, '%%@', must be doubled because the entire SELECT statement is inside a quoted string. LISTSERV will supply the quotes around the %1 parameter.

Using Dynamic Queries for access control

Once defined by the administrator, a dynamic query can be entered in most list header keywords used for access control or to enumerate e-mail addresses (see below for more information on security restrictions). Queries can be inserted wherever you would be allowed to enter an e-mail address, and have the following format:

Query(nickname[,parameter #1[,…]])

The nickname is mandatory and must refer to an existing, pre-defined dynamic query. Parameters are optional and must be enclosed in **true single quotes** (ASCII hex 27) if they contain white space or commas. **Do not use smart quotes or double quotes.** Smart quotes are not recognized by LISTSERV, and double quotes have a special meaning in list header keywords and are generally removed from the keyword.

Remember that dynamic queries can fail at run time, for instance because the LDAP server is not available or because the DBMS login credentials have expired. A dynamic query is not a suitable method for defining, for instance, the primary editor of a mailing list, but it is perfect for adding large groups of people as secondary editors.

Try to put the dynamic query last in access control keywords. LISTSERV will only execute the query if none of the prior access control patterns were a match. It is always more efficient for LISTSERV to match a pattern like '*@example.com' than to run a query against a DBMS or LDAP server.

Creating dynamic sub-lists

Dynamic queries can also be used as "virtual sub-lists" to augment the membership of a list (see below for more information on security restrictions). To do so, simply insert the query in the "Sub-Lists=" keyword, or create a "Sub-Lists=" keyword with just the query:

** Sub-Lists= PROJECT_A,Query(DEPT,HR),PROJECT_B*

Dynamic sub-lists work exactly like regular sub-lists, except that all recipients have the default list options. As always with sub-lists, duplicate recipients are eliminated.

Why not create plain dynamic lists instead?

It is not possible to create a regular list whose membership is defined by a dynamic query. Dynamic membership is only available by creating a super-list with a dynamic query as a sub-list, and the reasons for this design choice may not be obvious at first. In a nutshell, a plain dynamic list implementation would have led to reduced functionality for about the same learning curve. It may seem strange at first to define dynamic membership as a sub-list, but the only practical difference is the name of the list header keyword that carries the query and its parameters.

There are two main advantages to having the query in a sub-list rather than in the list itself. First, all the value-added LISTSERV features remain available – or rather, they can be made available if desired. A recipient from the dynamic query can subscribe to the list, change subscription options, be promoted to list editor, turn off mail receipt, and so forth. This would not be possible if the list membership were "frozen" and managed entirely outside of LISTSERV – for instance, in Active Directory. There is no function in Active Directory to switch to DIGEST mode or activate the NOMAIL option. This degree of freedom may not be desirable for internal lists and can be disabled easily with a list exit, or by preventing subscriptions to the super-list, but it can also be enabled. This provides more flexibility than an implementation where there is simply no way for a dynamic recipient to opt out.

Second, LISTSERV only explores sub-lists when it needs to, i.e. when a message is posted to the list, but explores and grooms top-level lists on a regular basis. Having the dynamic query as a sub-list minimizes the number of potentially resource-intensive queries. LISTSERV also knows that each sub-list has its own, independent list management process and will generally "do the right thing" with respect to managing, grooming and advertising the super-list vs. the sub-list. In a nutshell, LISTSERV will leave the sub-list recipients alone, as it always has when expanding sub-lists.

Creating a Dynamic Query exit to access custom data sources

LISTSERV supports a dynamic query exit to provide support for custom data sources other than LDAP and SQL databases, or to query LDAP or SQL data sources in a specific manner not otherwise supported by LISTSERV. This functionality is implemented as a standard LISTSERV exit with the following operational requirements:

- **Registration:** the name of the dynamic query exit is registered in the **DYN_QUERY_EXIT** configuration variable.

- **Input parameters:** LISTSERV converts the EXITPARM1 through EXITPARM4 keywords to plain-text counted format (see below) and concatenates them in order from 1 through 4 to form the exit input parameter. See sample decoding code below. *Warning: these keywords can contain arbitrary data. The exit script is responsible for escaping or removing harmful characters as required by its particular environment and programming language. These parameters should not be inserted 'as is' in shell command strings!*
- **Output requirements, successful completion:** if it completes successfully, the exit must return the standard success return code for the operating system on which LISTSERV is running, and create a result file (see below).
- **Output requirements, unsuccessful completion:** any error return code causes LISTSERV to fail the dynamic query. A result file does not need to be created in that case.

The result file

The exit script must store the results of the dynamic query in a plain-text result file named *exit.results*. LISTSERV deletes this file before calling the exit.

Each record in the result file must be in plain-text counted format (see below). The first record specifies a name for each of the attributes returned by the exit. This first record must be supplied even if the query returns no data. The second and following records contain attributes for each entry returned by the query, in the same order as the first record. See example below.

If the query succeeds, the exit must return at least an e-mail address for each entry, and it may return additional attributes. As with LDAP and DBMS queries, the name of the attribute containing the e-mail address is defined with the E-MAIL= keyword when configuring the query.

Plain-text counted format

A string is converted to plain-text counted format by prepending the length of the string, formatted as plain-text decimal characters, and an underscore. For instance, the string "Hello" becomes "5_Hello" and the empty string becomes "0_". Encoded strings are concatenated with no spaces or other delimiters in between. See example below.

Example

In this example, we will use a dynamic query called SAMPLE, which has been defined as follows:

TYPE=EXIT E-MAIL=MAIL EXITPARM1=Employees EXITPARM2=%1

The query is invoked as:

** Sub-Lists= Query(SAMPLE,HR)*

LISTSERV calls the exit with the following input argument:

```
9_Employees2_HR0_0_
```

The exit writes the following in the plain-text file, *exit.results*:

```
4_MAIL4_NAME
15_joe@example.com8_Joe User
16_jane@example.com9_Jane User
```

If the query had returned no results, the exit would have written the header record only :

```
4_MAIL4_NAME
```

The following sample REXX code (written for Windows exit conventions) illustrates the process of extracting exit parameters 1 through 4 from the exit input argument and creating a hardcoded query result:

```
 /* Extract EXITPARM1..4 */
 Parse arg data
 ep1 = Parm()
 ep2 = Parm()
 ep3 = Parm()
 ep4 = Parm()

 /* Create hardcoded result set to illustrate encoding process */
 ofile = 'exit.results'
 Call Lineout ofile,N('MAIL')N('NAME')N('PHONE')
 Call Lineout ofile,N('mike@example.com')N('Mike Smith')N('555-125-9182')
 Call Lineout ofile,N('marie@example.com')N('Marie Jourdain')N('N/A')

 /* Exit with SUCCESS return code (OS-dependent) */
 Exit 0

N:
 /* ENCODE to plain-text counted format */
 Procedure
 Parse arg line
 Return Length(line)'_'line

Parm:
 /* DECODE from plain-text counted format string 'data' (updated) */
 Procedure expose data
 Parse var data n'_'data
 If ^Datatype(n,'W') | n < 0 Then Exit 100 /* 100 = error on any OS */
 chunk = Left(data,n)
 data = Substr(data,n+1)
 Return chunk
```

Dynamic Queries and security

Only the LISTSERV administrator can define new dynamic queries. Once a query has been defined, list owners can use them:

- For access-control purposes, without any special authorization from the LISTSERV administrator. In an access-control scenario, the result of the dynamic query is a simple yes/no/error response. The data returned by the query is processed only to the extent necessary to determine if the attempted action is authorized or not.
- For enumeration purposes, if authorized by the LISTSERV administrator (see below). The result of an enumeration is a list of e-mail addresses that LISTSERV may process further, for instance

to send an administrative notice to the e-mail addresses enumerated in the "Notify=" list header keyword. In an enumeration scenario, LISTSERV only queries the e-mail and full name attributes of the directory (for DBMS queries, LISTSERV fetches all columns but discards all but the e-mail and name columns).

- To augment the membership of a mailing list, if authorized by the LISTSERV administrator (see below). In this scenario, all attributes are made available.

Note that there is no way for a list owner to obtain a list of available queries. List owners can only learn of the availability of pre-defined queries from the LISTSERV administrator. List owners are also unable to view the query definition.

To authorize the use of a dynamic query in a list header keyword, the LISTSERV administrator must update the list header the first time the query is used. From then on, the list owner can take of day-to-day list header management, and in particular:

- Update other keywords in the list header than the one containing the query.
- Change non-query elements in the keyword containing the query. For instance, if the "Notify=" keyword was changed to contain a query, the list owner can add additional e-mail addresses to the keyword.
- Change the relative position of the query within the keyword. For instance, change the relative position of a query in a "Sub-Lists=" keyword, where sub-lists are processed in the specified order.
- Remove the query altogether. In that case, only the LISTSERV administrator can re-add the query.

Authorization is on a per-keyword basis, and includes all parameters. A list owner authorized to use the DEPT query with parameter HR in the "Notify=" keyword can only use it in this keyword, and with the HR parameter.

Known issues and restrictions

The following known issues and restrictions exist:

- **Dynamic queries disable the special DBMS sub-list traverse feature.** When sending a posting to a super-list configured to use mail-merge for regular postings and whose sub-list recipients are hosted in a database ("DBMS= Yes" or equivalent in the list header), a special feature was activated in order to make additional DBMS columns available to the posting. Instead of processing the distribution normally, LISTSERV traversed the sub-list tree and processed the recipients one sub-list at a time, recognizing that each sub-list might have its own set of columns coming out of different tables with different names. This feature is disabled when one or more dynamic queries are used, because they cannot be tied to a physical sub-list. This does not affect mail-merge data extracted from the dynamic query itself. This is a delicate scenario in the

general case. L-Soft recommends not mixing “DBMS= Yes” and dynamic queries when mail-merge data must be extracted from both data stores.