Vitalware Documentation

Lookup List Maintenance

Document Version 1

Vitalware Version 2.3.01

www.kesoftware.com

Contents

Section 1

Overview

Lookup Lists have been a part of Vitalware since the first version of the software. They provide a useful mechanism for terminology control and are used extensively in the Vitalware client. The Lookup List facility stores data in a database table called eluts. The table contains a single record for each unique Lookup List entry. The information recorded for an entry includes:

`Name` — The name of the Lookup List. The name is used to associate a set of Lookup List entries with a particular field in the Windows client.

`Values` — A Lookup List entry may contain a number of values, with each value being a level in a hierarchy. The values start at level zero and increase. If a Lookup List is not part of a hierarchy (that is, it does not contain multiple levels), then only value zero is set. For hierarchies, a record will exist for each level in the hierarchy.

For example, if we have a Lookup List called Location, consisting of three levels:

- Country
- State
- City

with the data:

- `Australia` (Country)
- `Victoria` (State)
- `Bendigo` (City)

then three Lookup List entries (records) are generated:

	Entry 1	**Entry 2**	**Entry 3**
Country	`Australia`	`Australia`	`Australia`
State		`Victoria`	`Victoria`
City			`Bendigo`

The reason for three entries is that if a user wants to view a list of all countries (by viewing the *Countries* Lookup List for instance), then all Location Lookup entries that have the first value only filled are retrieved. As users may ask for any level in the hierarchy, the three records satisfy any potential request.

`Levels` — The number of levels filled for the current record.

Using the example above, *Entry 1* has a `Levels` value of 1 (as only a single level is filled), while *Entry 2* has a `Levels` value of 2, etc.

`Persistent`

If a Lookup List entry is not used anywhere in the system, the entry is removed from the Lookup List table. For example, if a record does not exist with a city location of `Bendigo`, then Entry 3 in the table above would be removed.

A `Persistent` Lookup List entry is not deleted if the entry is not used.

`Persistent` entries are used to pre-populate Lookup Lists with a known set of values, even if the values are not currently in use.

`Hidden`

If a Lookup List entry is marked as `Hidden`, then the entry is shown for searches but not when inserting new values.

For example, if the city `Bendigo` changed its name to `Sandhurst`, a new Lookup List entry is created with the city value `Sandhurt`. As the data still contains the value `Bendigo`, the Lookup List entry for this value is not removed. When users are inserting new records, we do not want the value `Bendigo` to appear in the Lookup List (as `Sandhurst` is the correct entry). However, when performing a search we still want `Bendigo` to appear in the list as there are still records containing this value. The `Hidden` attribute provides this functionality.

The `Hidden` setting is used to phase out entries that should no longer be used.

`Used`

The `Used` flag indicates whether the Lookup List entry is used anywhere within the system. If the flag is enabled, then at least one record in Vitalware uses the Lookup List entry's value.

`SortOrder`

By default, the entries in a Lookup List are displayed in alphabetic order. It is possible to display the entries in a user specified (i.e. customised) order. The `SortOrder` value is used as the sort key for sorting the values in a Lookup List with customised ordering. The value may be numeric or alphabetic, in which case a numeric or alphabetic sort is applied respectively.

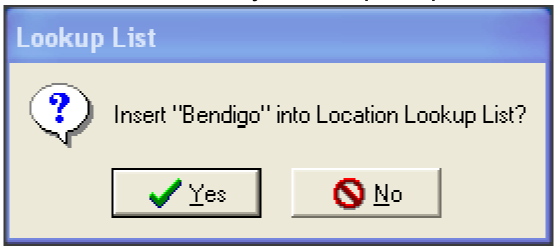
The `SortOrder` attribute is rarely used.

In order to implement customised sorting for a given Lookup List, the Windows client must be configured to provide the required functionality. Please contact KE Support for details.

"Dirty" Lookup Lists

There are a number of issues with the storage of Lookup List entries in the eluts database table:

1. Ideally, when deleting a record, any values in the deleted record which are in a Lookup List should be checked for uniqueness: if the values are not used in any other records, then the corresponding entry in the eluts database table should be deleted. However, the time required to perform these checks is prohibitive and in the interests of efficiency Vitalware does not perform them when a record is deleted. As a result Lookup List entries may exist in the eluts table where the value is not used in any records in the system.
2. If a user edits a Lookup value in a record and replaces it with a new value, when the record is saved they will be prompted to add the new value to the eluts table:

   
   Lookup List

   Insert "Bendigo" into Location Lookup List?

   However, the old value may not be used in other records in the system. As with the first issue, the time required to perform the check would dramatically increase the time required to save a record and, again, Vitalware does not perform the checks in the name of efficiency.

A consequence of these issues is that the eluts table can become "dirty", containing entries that are no longer required and which users should not be seeing.

To solve the problem of "dirty" Lookup Lists, Vitalware rebuilds the contents of the eluts table on a nightly basis (or as defined by the system maintenance schedule). The rebuild process may be quite time consuming for sites with large numbers of records. Since the rebuild process reads the Lookup List values for all records in Vitalware, it can build a new version of the eluts table containing only the correct entries. Once the rebuild is complete, the eluts table is back in sync with the Vitalware data.

The need to rebuild the Lookup List table each night means that the Lookup Lists are offline while the rebuild takes place. It also means that there is less time to perform other nightly maintenance routines (e.g. batch updates, etc.). Another issue is that since the contents of the eluts table are replaced each night, users cannot use records in the eluts table in the way they can for any other module. For a given entry it is not possible to:

- Add notes or multimedia.
- Follow audit trails on changes made.
- Set record level security.

It is also not possible to add new values to a Lookup List by simply adding a new record to the eluts table.

New Lookup Lists background service

To reduce the nightly system maintenance and to allow the Lookup List table to be used as a regular module, Vitalware 2.3.01 has added a background service that ensures the Lookup List entries are always in sync with the records in the Vitalware system. The addition of the service removes the need to rebuild the Lookup Lists on a nightly basis, hence reducing system maintenance time.

Furthermore, as the Lookup List table is no longer reloaded, the Lookup List module has been extended to include support for attributed notes (Notes tab) and multimedia (Multimedia tab). Audit trails on individual records are now maintained and Record Level Security may be used as for any other module.

Although a Lookup List rebuild program is still provided, it now applies changes to the eluts table rather than rebuilding the table completely. The introduction of the Lookup List service means that rebuilds are only required if the Lookup List table becomes corrupted, or if users accidentally delete records that are still in use.

lutserver

The server that handles updating of the eluts table in the new Lookup List service is called lutserver. lutserver runs on the Vitalware server machine waiting for requests to update Lookup List entries. The requests are generated by an audit trail plugin that picks up all deleted and modified records and determines what Lookup Lists need to be updated.

The process may be shown diagrammatically as:

```
      Audit Server (utils/auditserver)
                     ↓
  Lookup List Audit Plugin (etc/audit/luts.pl)
                     ↓
  Lookup List Service Data (loads/luts/data/*)
                     ↓
     Lookup List Server (utils/lutserver)
                     ↓
          Lookup List Table (eluts)
```

Let's use an example to describe the steps taken by the Lookup List service to ensure Lookup Lists are kept up to date. In this example a user changes the city value in a record from `Bendigo` to `Sandhurst` and saves the record. The following steps occur:

1. When the user saves the record, the Vitalware server generates an XML description of the changes made to the record. The Audit Server (auditserver) loads these changes and passes them on to all registered plugins. The Lookup List Services registers the Lookup List Audit Plugin when the Audit Server is started. For our example record, the changes will show that `Bendigo` was changed to `Sandhurt` in the *City* column.
2. The Lookup List Audit Plugin looks at the changes supplied by the Audit Server. For each column changed it checks to see if a Lookup List is associated with the column. For our example, it will determine that the *City* column is associated with the *Location* Lookup List. It will then look at the two values and their associated operations. The value `Bendigo` has a delete operation (since it is being removed) and the value `Sandhurt` has an insert operation (as it is the new value). Since the Windows client was used to save the record, the Audit Plugin will ignore the insert operation as the user will have been asked to add the new entry to the *Location* Lookup List if it did not exist already. The delete operation is passed to the Lookup List Server for processing.
3. The Lookup List Audit Plugin writes a file containing the:
   - Operation performed (delete)
   - Value deleted (`Bendigo`)
   - Lookup List Name (*Location*)
   - Column changed (*City*)

   The file is located in the `loads/luts/data` directory. The format of the file name is `date.time.table.irn`, where:

   - `date` is an eight digit date in `yyyymmdd` format.
   - `time` is a six digit time in `hhmmss` format.
   - `table` is the database table in which the record was modified.
   - `irn` is the key number of the record modified.

   An example file name is: `20121026.133308.epos.375`.

4. The Lookup List Server fetches all files in the `loads/luts/data` directory and sorts them into date/time order. It is important that the files are processed in the same order as they were created, otherwise synchronisation issues may arise. The Lookup List Server reads the contents of each file, extracting the information within. Then for each Lookup List in the file it determines the operation to apply (insert or delete):
    - For an insert operation it checks to see if the value is already in the Lookup List; if not it inserts a new value into the eluts table.
    - For a delete operation it checks whether the value is used anywhere in the Vitalware system for the given Lookup List name. If the value is not used, the record is deleted from the eluts table (provided it is not `Persistent`).

    Once the file has been processed, it is removed. Once all the files have been processed, the Lookup List Server waits for new files to process.

The reason for so many steps is that audit plugins must be fast so that the Audit Server can process audit records quickly. To ensure the audit plugin is fast, the main Lookup List processing is removed from the audit plugin and moved to the Lookup List Server. The split ensures that the Audit Server keeps up with audit changes, even when the Lookup List Server may lag.

The Lookup List Server handles all deletion operations, that is it checks whether a Lookup List entry is still in use and if not, deletes it. It also handles insertions (that is new values) from all sources except the Windows client.

The Lookup List Server and the Registry

The Lookup List Server complies with the Lookup and Lookup Exact Registry entries:

- If a column has Lookup Exact set to `true`, then all comparisons with existing entries in the Lookup List table are performed as exact matches, i.e. the character case and punctuation must match exactly.
- If Lookup Exact is not enabled, all punctuation and character case is ignored for value comparisons. Search the Vitalware Help for details on the `Lookup Exact Registry entry` for more details.

The Lookup Registry entry is used to control the flags set when adding new values to a Lookup List. The table below lists each setting and examines how the Lookup List Server implements the required functionality. The Lookup Registry entry settings are generally set on a per column basis.

Setting	Description
`skip` `readonly`	The entry will not be added to the Lookup List table. If the column is part of a hierarchy, the entry is only skipped if the bottom level has this setting enabled. For example, if the *State* column has skip enabled, then entries with *Country*, *State* and *City* will be created, but entries with just *Country* and *State* will be skipped.
`readwrite`	The `readwrite` setting adds a new value to the Lookup List table. If an entry already exists but has the `Hidden` flag enabled, the flag will be reset (i.e. disabled). If the `Used` flag is disabled, it is enabled.
`autowrite`	The `autowrite` setting adds a new value to the Lookup List table. If an entry already exists and the `Used` flag is disabled, it is enabled.

`autowriteignore`
`readignore`
`writeignore`
	If the entry does not exist, a new entry is created with `Hidden` enabled. If an entry already exists and the `Used` flag is disabled, it is enabled.

Finally, the Lookup List Server will not delete any entry from the Lookup List table that has `Persistent` enabled.

The Lookup List Server, lutserver, provides the functionality required to keep the Lookup List table in sync with values used within the Vitalware System. The addition of the server removes the need to rebuild the Lookup List tables on a nightly (or otherwise) basis.

vwlutsrebuild

Prior to Vitalware 2.3.01, `vwlutsrebuild` was used to rebuild the Lookup List tables on a nightly basis. Functionality added with Vitalware 2.3.01 means that Lookup List tables no longer need to be rebuilt and although in theory `vwlutsrebuild` is no longer required, there may be occasions when the Lookup List tables need to be updated manually. It would be useful for example to be able to regenerate a missing entry if someone deleted a record accidentally.

The `vwlutsrebuild` server side program has been rewritten to work with the Lookup List Server framework (introduced with Vitalware 2.3.01). The program no longer generates a data file with all Lookup List entries in it that is then loaded into an empty Lookup List table. Instead, the program now visits each Vitalware record and checks that the values in the records are in the Lookup List table. If a value is not present, a new entry is created. Any entries that are no longer used, are deleted unless they are `Persistent`.

The rebuild process occurs in two stages. The first stage involves generating a list of all changes that need to be applied to the Lookup List table. This stage is known as the data generation phase. There are three types of changes possible:

- *insert* - new records to be added to the Lookup List table.
- *update* - existing records that require their flags (`Hidden`, `Used`, `Persistent`) to be adjusted.
- *delete* - existing records that are no longer required.

The data for each of the above changes is placed in the `luts` directory in files named `data.insert`, `data.update` and `data.delete` respectively.

Once the data generation phase is complete, the loading phase is executed. In this phase the files generated are loaded into the Lookup List table. Once the load is finished, the Lookup List table rebuild is complete.

The usage message for `vwlutsrebuild` is:

```
Usage: vwlutsrebuild [-dlmqtv] [lookup list ...]

-d    only produce lookup table data loading files

-f    load lookup changes quickly (takes eluts offline)

-l    only load lookup table data file

-t    include lookup list text files in rebuild

-v    verbose mode, print debugging information
```

The following table describes each option:

Option	Description
`-d`	Run the data generation phase only: the data files to be loaded are generated but not applied.
`-l`	Skip the data generation phase and run the loading phase only. The data files to be loaded are assumed to exist.
`-f`	Determine whether the Lookup List table should be taken offline while the loading phase is underway. If the `-f` option is specified, the `eluts` table will be taken offline and the data loaded, otherwise the table will be left online while the load proceeds. The load will be much faster for large amounts of data if the `-f` option is specified.
`-v`	Useful for tracing what the data generation phase is producing. Each entry to be inserted, updated or deleted is printed to the screen (as well as added to the appropriate data files).
`-t`	Informs `vwlutsrebuild` to include text files found in the `luts/defaults` and the `local/luts/defaults` directories as part of the data generation phase. The text files are used to load pre-built Lookup List values. Entries found in these files are loaded with `Persistent` enabled. The format of the text file is:

```
#
# Lines beginning with a hash character are comments
#
[-]lookup list name|value 1[|value 2|...][=sortorder]
```

where:

lookup list name	is the name of the Lookup List into which the value is to be loaded.
Value 1	is the value for the first level of the Lookup List. For hierarchies, the successive levels are separated by a pipe (|) symbol.
sortorder	An optional equals sign with a sort order may be used for Lookup Lists that support customised ordering. The *sortorder* value is added to the `SortOrder` entry for the Lookup List. A leading minus sign disables loading of the Lookup List, i.e. all values for that list will be skipped.

Where a hierarchy entry is specified, it is not necessary to add the upper levels of the hierarchy as these are added automatically. For example, if a text file contained:

```
Location|Australia|Victoria|Bendigo
```

then three entries are generated. The entries are the same as specifying:

```
Location|Australia

Location|Australia|Victoria

Location|Australia|Victoria|Bendigo
```

The use of text files provides a convenient mechanism for pre-loading Lookup Lists. As stated above, each entry loaded will have `Persistent` enabled, meaning the entry will persist even if it is not used. The only way to delete such an entry is to locate the record in the Lookup List module and delete it manually.

A System Administrator may configure a table to be skipped when Lookup Lists are being rebuilt. If a line of the form:

```
RELUTS=no
```

is found in a file called `vwoptions` in the database directory, `vwlutsrebuild` will not check data in that table.

The arguments to `vwlutsrebuild` is a list of Lookup List names to rebuild. If a list is not supplied, then all Lookup Lists are checked. The following examples cover some of the uses of `vwlutsrebuild`:

Rebuild all Lookup Lists

The command used to rebuild all Lookup Lists is:

```
vwlutsrebuild -t
```

It should not be necessary to run this command. The Lookup List Server maintains Lookup List synchronisation. The command may be used to ensure the Lookup List table is synchronised correctly.

Check Lookup List table is synchronised

The command used to check whether the Lookup List table is synchronised is:

```
vwlutsrebuild -d -t -v
```

This command runs the data generation phase only. The Lookup List table is not modified. The verbose option prints out any inconsistencies found.

Rebuild the Location and Admin Names Lookup List

The command used to rebuild the `Location` and `Admin Names` Lookup Lists is:

```
vwlutsrebuild Location 'Admin Names'
```

This command rebuilds the `Location` and `Admin Names` Lookup Lists. As the `-t` option is not specified, any entries in text files will not be loaded.

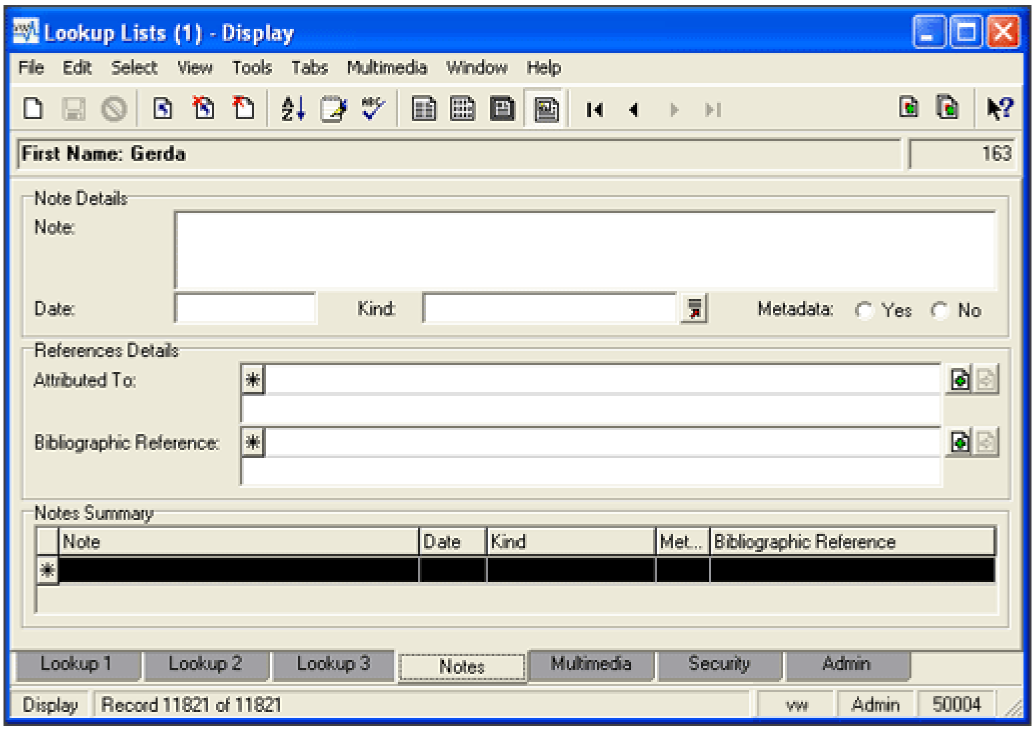
Note the use of single quotes to enclose the names of Lookup Lists that contain spaces.

Lookup List module

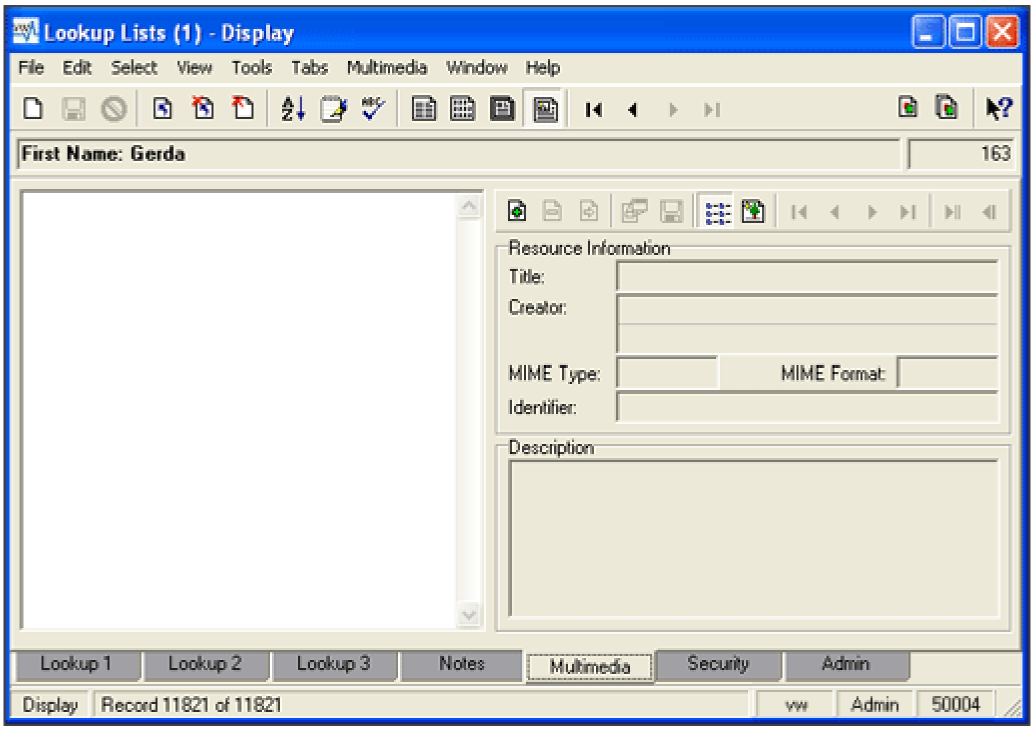
Until the release of the new Lookup List maintenance facilities in Vitalware 2.3.01, users could not interact with the Lookup List module and expect their changes to be preserved. The module itself was simplified to the point where only Lookup List specific information was stored. Fields available in all other modules were disabled. The maintenance changes introduced with Vitalware 2.3.01 mean that users can now access and use the module as they would any other module.

The following tabs are now available and functional in the Lookup Lists module:

Notes



Multimedia

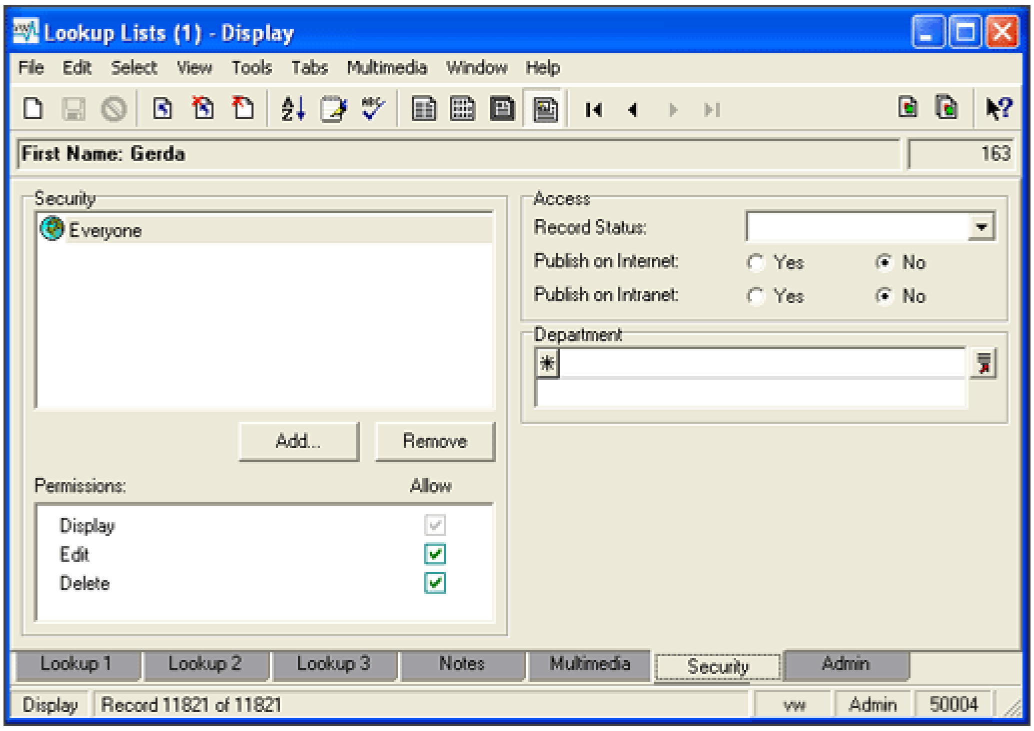

Lookup Lists (1) - Display
File Edit Select View Tools Tabs Multimedia Window Help
Lookup 1
Lookup 2
Lookup 3
Notes
Multimedia
Security
Admin

First Name: Gerda 163

Resource Information
Title:
Creator:
MIME Type: MIME Format:
Identifier:
Description

Display | Record 11821 of 11821 | vw | Admin | 50004

Security


Lookup Lists (1) - Display
File Edit Select View Tools Tabs Multimedia Window Help
Lookup 1
Lookup 2
Lookup 3
Notes
Multimedia
Security
Admin
Add...
Remove

First Name: Gerda 163

Security
Everyone

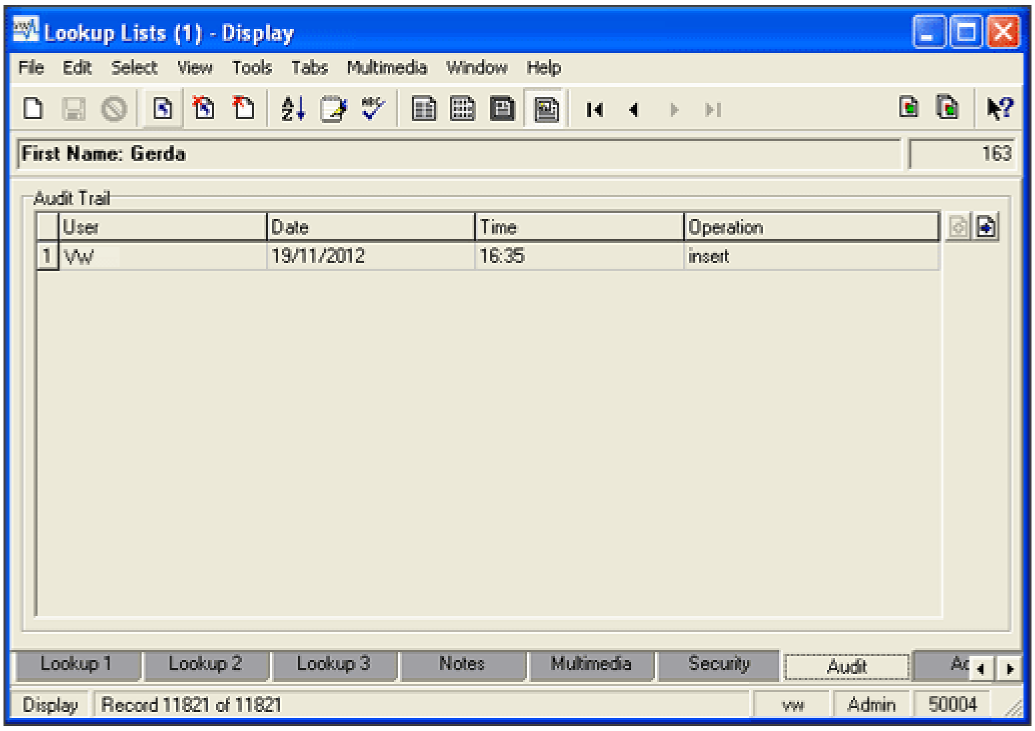
Access
Record Status:
Publish on Internet: Yes [✓ No]
Publish on Intranet: Yes [✓ No]

Department

Permissions:	Allow
Display	✓
Edit	✓
Delete	✓

Display | Record 11821 of 11821 | vw | Admin | 50004

Audit


Lookup Lists (1) - Display
File Edit Select View Tools Tabs Multimedia Window Help

First Name: Gerda 163

Audit Trail

	User	Date	Time	Operation
1	VW	19/11/2012	16:35	insert

Lookup 1
Lookup 2
Lookup 3
Notes
Multimedia
Security
Audit

Display | Record 11821 of 11821 | vw | Admin | 50004

SECTION 2

Conclusion

The changes to the Lookup List maintenance framework made in Vitalware 2.3.01 provide a number of benefits:

- Lookup Lists are always up to date. Once the last use of a Lookup List value is deleted, the entry is removed from the Lookup List table.
- The Lookup List nightly maintenance procedure is no longer required. The removal of the maintenance decreases the amount of time required by the nightly maintenance routines.
- The Lookup List module now provides the standard tabs available in all other Vitalware modules.
- The `vwlutsrebuild` command may be used to check the consistency of the Lookup List table and apply any needed adjustments.

The changes to the Lookup List maintenance mechanism are the first of a series of changes designed to decrease the amount of time Vitalware requires to complete its maintenance runs.

Index