VMS Software

VSI OpenVMS

VSI OMNI API Omni Directory Services User Guide

Document Number: DO-DDWINS-01A

Publication Date: April 2024

Operating System and Version: VSI OpenVMS IA-64 Version 8.4-1H1 or higher
VSI OpenVMS Alpha Version 8.4-2L1 or higher

Software Version: VSI OMNI Version 4.1

VMS Software, Inc. (VSI)
Boston, Massachusetts, USA

VSI OMNI API Omni Directory Services User Guide

Legal Notice

Preface

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended Audience

This document is for application programmers, network managers, and system managers who have experience in network management and distributed processing.

3. Document Structure

The manual is organized as follows:

- Chapter 1: *Using the Omni Directory Services*
- Chapter 2: *Omni Directory Services Command Language*
- Chapter 3: *Command Descriptions*
- Appendix A: *Online Help*

4. Associated VSI OMNI Documents

The following documents provide more information about using the VSI OMNI API software:

- *VSI OMNI API Guide to Using OmniView*
- *VSI OMNI API Omni Definition Facility User Guide*

5. Related ISO/IEC Documents

The following documents provide more information about the ISO/IEC standard Manufacturing Message Specification (MMS):

- *Industrial Automation Systems — Manufacturing Message Specification Service Definition, ISO/IEC 9506–1*
- *Industrial Automation Systems — Manufacturing Message Specification Protocol Specification, ISO/IEC 9506–2*

6. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

7. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: <docinfo@vmssoftware.com>. Users who have VSI OpenVMS support contracts through VSI can contact <support@vmssoftware.com> for help with this product.

8. Conventions

The following conventions are also used in this manual:

Convention	Meaning
VSI OMNI	The term "VSI OMNI" refers to the VSI OMNI API product or to functions and services provided by the VSI OMNI API software.
OpenVMS	The term "OpenVMS" refers to OpenVMS VAX or OpenVMS Alpha products, or to operations and functions performed by the OpenVMS VAX or OpenVMS Alpha operating system.
$	The dollar sign is the default OpenVMS system prompt for user input.
ODSCL>	The ODSCL acronym with a right angle bracket is the Omni Directory Services Command Language prompt for user input.
omni_commands	VSI OMNI commands follow the conventions and requirements for C programming. The commands are case-sensitive and must be entered exactly as shown.
UPPERCASE, lowercase	The system differentiates between uppercase and lowercase characters. Literal strings that appear in descriptions, examples, or command syntax must be entered exactly as shown.
Boldface type	Boldface type emphasizes user input to system prompts.
`system output`	This typeface indicates system output in interactive examples.
[ ]	Square brackets are part of the directory specification `[directory-name]` on OpenVMS systems. Square brackets in a procedure call indicate parts of a parameter that can be included or omitted.
Ctrl/ *x*	Hold down the **Ctrl** key while you press another key, indicated here by *x*.
italic type	Italic type emphasizes important information, names of API calls and procedures, or the complete titles of documents.
. . .	Vertical ellipses (dots) in examples indicate that information has been omitted for clarity.

Chapter 1. Using the Omni Directory Services

This chapter provides an overview of the Omni Directory Services (ODS) facility. ODS is a file-based directory service for the storage, management, and retrieval of information objects.

1.1. Database Data

ODS maintains two databases, the directory database and schema database. The directory database stores information about objects. The schema database contains meta-data, which is data that defines and gives meaning to the information in the directory database.

1.1.1. Directory Data

ODS directory data stores information about real-world objects. The information for an object is called either an entry or directory entry. An entry in the directory database contains information about an object and is not the object itself.

These objects are stored in cache files in `ods:[cache]`.

1.1.2. Schema Data

ODS schema data defines object classes and attribute types. This data appears in the files, `ods_known_attribute_types.dat, ods_known_object_classes.dat`, and in `ods:[local]`:

- An object class is defined by a unique object class abbreviation in the object class schema file.
- An attribute type is defined by a unique attribute type abbreviation in the attribute types file.

Object Classes

Object classes are categories of objects and every object belongs to an object class. The object class determines what attributes a member object requires and also what additional attributes a member object can possess.

All object classes mandate an attribute called object class that specifies the object class to which the object belongs. An object is thus an unordered collection of attributes that are textually shown separated by slashes to delimit the beginning and end of each type-value pair. For example:

```
/OC=TAE/P_ADDR=A.B.C.D/APC={2 5 9 1}
```

The `handle` or `key` to an object is its distinguished name or name. The name is the means by which an object is accessed.

The name of an object is a sequence (an ordered list) of naming attributes (attributes whose types are flagged “naming” in the schema). The last attribute in the sequence belongs to the named object and is known as the Relative Distinguished Name (RDN).

In a flat naming convention, the RDN is sufficient for constructing a distinguished name. In a hierarchical naming convention, the RDN of the object can be preceded by the RDNs of objects superior to the object being named in a tree-structured directory.

Consider the following object class:

```
ObjectClass EXAMPLE
{
mustContain : OC, CN (naming), P_ADDR
{
mayContain : APT, AEQ, APC
}
```

An object X of object class EXAMPLE can be defined in ODS according to a flat naming convention:

```
NAME: /CN=X
ENTRY: /OC=EXAMPLE/P_ADDR=0x01.0x01.0x01.LOCAL/APC={2 1 2}
```

It can also be defined in ODS according to a hierarchical naming convention:

```
NAME: /O= /OU=ENGINEERING/CN=X
ENTRY: /OC=EXAMPLE/P_ADDR=0x01.0x01.0x01.LOCAL/APC={2 1 2}
```

Attribute Types

An attribute is a type-value pair, consisting of both a type and value. The type of the attribute determines the following:

- The syntax of the attribute value
- Whether the attribute can be used in naming the object
- A default value (if any)
- The upperbound of the attribute value

An attribute is textually represented as:

`Attribute_type_abbreviation=Attribute_value`

For example:

`OC=VMD`

1.2. Naming Conventions

Each ODS object must have an unambiguous name that applies only to that object. However, a single object can have aliases (alternate names) in addition to its unambiguous name.

An alias is an object of the object class ALIAS with the mandatory attribute ALIASED_NAME that is set to the name of an object. For example:

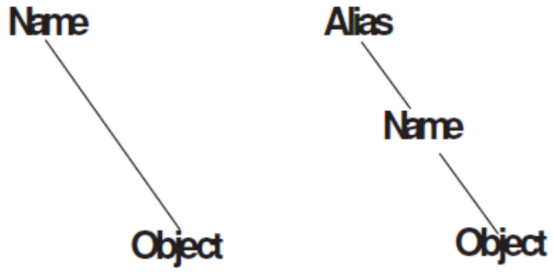
```
NAME: /CN=Alternate_X
ENTRY: /OC=ALIAS/ALIASED_NAME=(/CN=X)
```

Here, the name `/CN=Alternate_X` accesses the same object that `/CN=X` does. This is called *dereferencing* an alias.

ODS supports multiple levels of aliasing — that is, an alias can reference another alias, and so on. The alias name points to the object's unambiguous name, which in turn points to the object.

Figure 1.1, “ODS Name and Object Relationship” shows the directory path between a name and an object, and between an alias, name, and object in the directory.

Figure 1.1. ODS Name and Object Relationship



Each object belongs to an object class — an identifiable family of objects that have common characteristics. For example, Country, Organization, Person, and Application process are object classes. However, John Smith and Jane Doe, both of whom possess the common characteristics of a first name and a surname, are objects in the Person object class.

For each object, there is a collection of attributes that describe the characteristics of the object. Each attribute has a type and one value.

An object has both naming and entry attributes. A naming attribute names an object. An entry attribute specifies characteristics that the object possesses. For example, you can specify the following during a REGISTER operation:

```
ODSCL> reg dir name "/ou=CMPD/sy=UNTNTD/cn=XTI Responder" -
               attr "/oc=TAE/p_addr=..0x4900ccaa0004002f2c"
```

In this example, the naming attributes are:

```
/ou=CMPD/sy=UNTNTD/cn=XTI Responder
```

The entry attributes are:

```
/oc=TAE/p_addr=..0x4900ccaa0004002f2c
```

1.3. Accessing the Omni Directory Services Facility and Data

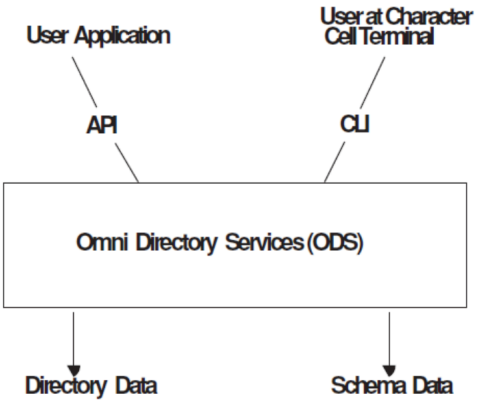
You can access ODS using the VSI OMNI Application Program Interface (API) routines, the ODS Command Line Interface (CLI), or the command `run ods:[exe]odscl`.

Multiple users can access ODS using any of the following methods:

- API ODS routines
- ODS Command Language (ODSCL)

Figure 1.2, “ODS Access Paths” shows the paths you can follow to access ODS and to store or retrieve ODS information.

Figure 1.2. ODS Access Paths



Chapter 2. Omni Directory Services Command Language

The Omni Directory Services Command Language (ODSCL) provides a set of commands that enable users to execute Omni Directory Services (ODS) functions.

Using ODSCL, you can list the contents of the directory, then read, add, remove, or modify an entry in that directory. ODSCL allows multiple users to simultaneously execute ODS directory service functions.

ODSCL interacts with the ODS routines to perform the directory service functions. ODS is modeled after ISO International Standard 9594 for the Open Systems Interface (OSI) Directory.

The ODSCL commands are as follows:

- DEREGISTER DIRECTORY deletes an object from the directory.
- EXIT exits from ODSCL and returns you to the default system prompt.
- HELP displays online information about a particular topic or subtopic and provides informational text about the ODSCL commands.
- LIST DIRECTORY displays objects in the directory that match either the specified selection criteria or all entries in the directory.
- MODIFY DIRECTORY modifies an object in the directory by adding or deleting an attribute or by changing an attribute value.
- READ DIRECTORY displays information about an object in the directory.
- REGISTER DIRECTORY creates an object in the directory.
- SET DIRECTORY establishes a name directory path used to reference the name of an object based on its relative position in the name hierarchy.
- SHOW DIRECTORY displays the current name directory path.

2.1. Command Syntax

The ODSCL command string requires the following elements in the order shown:

1. A verb that specifies the operation to execute
2. An entity that specifies what the verb acts upon
3. One or more attributes (or arguments) that place boundaries on the operation specified by the verb
4. Optional qualifiers that specify how the operation is to be handled

Most ODSCL commands contain a verb, entity, and attribute. However, EXIT and HELP do not require an entity or attribute and LIST does not require an attribute.

Example 2.1. ODSCL Command Syntax

```
Verb Entity Attribute <=> "attribute_expression"
```

```
            <Argument <=> "argument_expression">
            <Qualifier <=> "qualifier_expression">
For example:
READ DIRECTORY NAME "/o=cmpd/sy=test"
(verb) (entity) (attribute) (attribute name expression)
```

Example 2.1, “ODSCL Command Syntax” shows the generic ODSCL command line syntax with an example.

Usage Notes

The following are ODSCL command guidelines:

- A word enclosed by angle brackets < > is optional.
- An ODSCL verb is one word long and may require attributes or arguments.
- The valid ODSCL entity is DIRECTORY.
- ODSCL attributes consist of an attribute keyword and an attribute expression. Valid attribute keywords are NAME, ATTRIBUTES, and PATH.

  The attribute expression consists of a string of one or more attribute abbreviations and attribute values, delimited by a slash. The entire attribute expression is enclosed in double quotes. For example:

  ```
  attributes "oc=ae/p_addr={1 2 3}"
  ```

  The attribute expression for an object name must also include either a leading slash in the attribute expression that refers to the object’s absolute name in the hierarchical structure, or a tilde (~) that refers to the relative name in the structure.

  The attribute expression for the name keyword refers to the object’s absolute name in the hierarchical structure. For example:

  To reference an absolute name: `name "/o=cmpd/sy=te"`

  To reference a relative name: `name "~/sy=te"`

- Arguments modify the specific operation that the command performs. Enter arguments using an argument keyword and argument expression. Valid argument keywords are ADD, REMOVE, and REPLACE:

  - For the ADD argument keyword, the argument expression consists of a string of one or more attribute abbreviations (as defined in the known attribute types file) and attribute values to be added, each delimited by a slash. The entire argument expression is enclosed with double quotes. For example:

    ```
    add "ou=test"
    ```

  - For the REMOVE argument keyword, the argument expression consists of a string of one or more attribute abbreviations (as defined in the known attribute types file) to be deleted, each delimited by a slash. The entire argument expression is enclosed with double quotes. For example:

    ```
    remove "p_addr/l"
    ```

- For the REPLACE argument keyword, the argument expression consists of one or more attribute abbreviations, the old attribute and the new attribute value separated by a comma. The entire argument expression is enclosed with double quotes. For example:

  ```
  replace "ou=test,newtest"
  ```

- Qualifiers affect the way a command is processed. The qualifier requires a qualifier expression that specifies how the command is to be qualified. The valid qualifier keywords are: MATCH, WITH, and TO FILE.

  - For the MATCH qualifier keyword, the qualifier expression consists of a string of one or more attribute abbreviations (as defined in the known attribute types file) and attribute values to be matched, delimited by a space ampersand space (&). The entire qualifier expression is enclosed in brackets. For example:

    ```
    MATCH [oc=cmpd & sy=test]
    ```

  - For the WITH qualifier keyword, the qualifier expression consists of a string of one or more operating options and an option state (on or off) for operating options to apply for this command, delimited by a slash. The entire qualifier expression is enclosed with double quotes. For example:

    ```
    WITH ‘‘print=on’’
    ```

  - For the TO FILE qualifier keyword, the qualifier expression contains the name of the file into which the output will be written. For example:

    ```
    TO FILE test.t
    ```

2.2. Command Features

ODSCL helps you to enter ODSCL commands by providing the following features:

- You can abbreviate any command to the minimum number of letters that make it unique, usually three letters. The parser returns an error message if the term is ambiguous.
- ODSCL is case-insensitive and allows you to run a single command without remaining in ODSCL.
- ODSCL steps you through the correct syntax and provides a list of options for verbs, entities, attribute keywords, argument keywords, qualifier keywords,and valid attribute abbreviations. If you do not supply all of the required elements, command completion prompting reminds you to supply information for each missing element, one element at a time.
- If unique words are encountered in the command line, it is not necessary to enter remaining keywords or attributes to execute the command. The CLI includes the missing words.
- The hyphen (-) can be used as a continuation character. To continue a command line, enter a hyphen as the last character on the line and press Return. ODSCL stores the line and displays the continuation prompt _ODSCL> so you can enter the rest of the command.
- ODSCL displays data in a three-part form that consists of a header, text message, and data area. The header displays information about the request. The text message consists of an optional message about the status of the request. The data area contains the actual data returned in response to the request.

Example 2.2, “ODSCL Display” shows the display for a READ operation.

Example 2.2. Example 2.2. ODSCL Display

```
ODSCL>  read directory name "/o=test/ou=test/cn=test"

ODSCL   (Read Directory)          | <-- Header
On Tuesday 4-Jul-17 13:42:41      |

Information about name:  /o=test/ou=test/cn=test  |  <--Text
Displayed entry attribute count: 3                |     Message

Attribute     Attribute             Attribute
Abbreviation  Description           Value
OC            Object Class          AE(Application Entity)    | <-- Data
APC           Application Context   TEST                      |     Area
P_ADDR        Presentation Address  TEST
ODSCL>
```

2.3. Invoking and Exiting from the Command Language

When ODSCL starts, there are a number of operating options that can be specified. When specified, the default operating state of the option is modified for the current execution of ODSCL. If ODSCL starts with no specified options. the default operating parameters apply as indicated below. Use a foreign command to invoke ODSCL.

```
odscl -option -option ... OR odscl -options
```

The following list summarizes the valid operating options and default states:

- –p — Print mode for ODS option. The default state is on, which causes ODS to print completion error status messages on the standard error device.
- –s — Default schema information only option. The default state is off, which causes ODS to use the user-defined schema information in addition to the default schema information.
- –c — Schema consistency check option. The default state is on, which causes ODS to check the dynamic schema information for consistency.
- –a — Alias dereferencing option. The default state is on, which causes aliases to be dereferenced when reading or listing directory information.
- –e — Exact match for the list command option. The default state is off, which causes a case-insensitive selection match in the list command.

Enter the EXIT command to exit from ODSCL. The EXIT command causes a clean shutdown of ODSCL. Otherwise, you can press **Ctrl /Z**.

2.4. Obtaining Online Help on Commands

After invoking ODSCL, you can use the HELP command to obtain help for ODSCL topics and subtopics. You can also obtain help about your current command by entering the word HELP or a question mark (?).

Chapter 3. Command Descriptions

This chapter describes the Omni Directory Services Command Language (ODSCL) commands:

- DEREGISTER DIRECTORY NAME
- EXIT
- LIST DIRECTORY
- MODIFY DIRECTORY NAME
- READ DIRECTORY NAME
- REGISTER DIRECTORY NAME
- SET DIRECTORY PATH
- SHOW DIRECTORY PATH

DEREGISTER DIRECTORY NAME

DEREGISTER DIRECTORY NAME — Deletes a specified object name from the directory.

Syntax

```
LIST DIRECTORY  [MATCH [attr_abbrev=value & attr_abbrev=value . . . ]]
                [WITH "operating_option=state"]
                [TO FILE file_name]
```

Attributes

`DIRECTORY`

DIRECTORY is the entity on which the DEREGISTER command operates.

`NAME "/attr_abbrev=value/attr_abbrev=value . . . "`

The quoted attribute name expression that defines the name of the object in the directory hierarchy. *attr_abbrev* is the name attribute abbreviation as defined in the known attribute types file.

`WITH "operating_option=state"`

Identifies the operating option and its state (on or off) to use for this operation. If WITH "operating_option=state" is not specified, operating options specified when ODSCL was invoked apply; or if operating options are not specified at invocation,default operating parameters apply.

EXIT

EXIT — Shuts down ODSCL and returns the user to the system prompt.

Format

EXIT

HELP

HELP — Displays informational text about the current ODSCL prompt when entered during an ODSCL command. If entered at the ODSCL> prompt, HELP displays information about a particular topic or subtopic as defined in the HELP hierarchy.

Syntax

```
HELP   <Topic-keyword>
       [Subtopc-keyword]
```

Attributes

Topic-keyword

The ODSCL topic about which information is requested. See Appendix A, *Online Help* for a listing of valid help topic and subtopic keywords.

Subtopic-keyword

The ODSCL subtopic about which information is requested. See Appendix A, *Online Help* for a listing of valid help topic and subtopic keywords.

Usage Notes

At the ODSCL prompt ODSCL>, HELP displays information about the requested topic or subtopic. A list of additional topics or subtopics is also displayed. The user is prompted to enter another topic or subtopic. Pressing Return at the current prompt returns you to the next higher-level prompt.

LIST DIRECTORY

LIST DIRECTORY — The LIST command lists all name entries and associated attributes, or selective entries and associated attributes matching the selection filter in the directory

Syntax

```
LIST DIRECTORY   [MATCH [attr_abbrev=value & attr_abbrev=value . . . ]]
                 [WITH "operating_option=state"]
                 [TO FILE file_name]
```

Attributes

DIRECTORY

Indicates the entity on which the LIST operation is being performed.

MATCH attr_abbrev=value & attr_abbrev=value . . .

The optional set of attribute abbreviation and values to match. *attr_abbrev* is the attribute abbreviation as defined in the known attribute types file.

WITH "operating_option=state"

Identifies the operating option and its state (on or off) to use for this operation. If WITH "operating_option=state" is not specified, operating options specified when ODSCL was invoked apply; or if operating options are not specified at invocation,default operating parameters apply.

TO FILE file_name

Redirects output to a new file or appends output to an existing file.

Usage Notes

The MATCH guidelines are as follows:

- The MATCH expression consists of an attribute abbreviation and MATCH value. The ampersand (&) character signifies a logical AND.
- The MATCH can be performed in a non-case-sensitive fashion (the default state for the operating option) or in a case-sensitive fashion, based on the state of the operating option.

MODIFY DIRECTORY NAME

MODIFY DIRECTORY NAME — Modifies attributes and attribute values associated with a specified object name in the directory. Attributes can be added, changed, or removed.

Syntax

```
MODIFY DIR NAME   "/attr_abbrev=value/attr_abbrev=value . . . "
                  [ADD "attr_abbrev=value/attr_abbrev=value . . . "]
                  [REMOVE "attr_abbrev=value/attr_abbrev=value . . . "]
                  [REPLACE "attr_abbrev=oldvalue, newvalue/
                           attr_abbrev=oldvalue . . . "]
                  [WITH "operating_option=state"]
```

Attributes

DIRECTORY

Indicates the entity on which the MODIFY operation is being performed.

NAME "/attr_abbrev=value/attr_abbrev=value . . . "

The quoted attribute name expression that defines the name of the object in the directory hierarchy. *attr_abbrev* is the name attribute abbreviation as defined in the known attribute types file.

ADD "attr_abbrev=value/attr_abbrev=value"

The quoted argument expression that defines the attributes and values to add. *attr_abbrev* is the attribute abbreviation as defined in the known attribute types file.

REMOVE "attr_abbrev=value/attr_abbrev=value . . . "

The quoted argument expression that defines the attributes to remove. *attr_abbrev* is the attribute abbreviation as defined in the known attribute types file.

REPLACE "attr_abbrev=oldvalue, newvalue/attr_abbrev=oldvalue, newvalue . . . "

The quoted argument expression that defines the attributes to change with their old and new values. *attr_abbrev*

is the attribute abbreviation as defined in the known attribute types file.

WITH "operating_option=state"

Identifies the operating option and its state (on or off) to use for this operation. If WITH "operating_option=state" is not specified, operating options specified when ODSCL was invoked apply; or if operating options are not specified at invocation,default operating parameters apply.

Usage Notes

The following guidelines apply:

- The object class attribute (the attribute abbreviation OC) and name attributes may NOT be modified.
- This command supports the addition and removal of optional attributes. Both optional and mandatory attributes may be modified.
- If the disable consistency check operating parameter is on (the default state), attribute modifications are checked for consistency. The supplied attributes of the object are checked against the definition of the object class in ods_known_object_classes.dat. The operation fails if an inconsistency exists.
- If the operation is completed successfully, the local directory information is updated with the new information.

READ DIRECTORY NAME

READ DIRECTORY NAME — Displays attributes associated with a specified object name in the directory.

Syntax

```
READ DIRECTORY NAME  "/attr_abbrev=value/attr_abbrev=value . . . "
                     [WITH "operating_option=state"]
```

Attributes

DIRECTORY

Indicates the entity on which the READ operation is being performed.

NAME "/attr_abbrev=value/attr_abbrev=value . . . "

The quoted attribute name expression that defines the name of the object in the directory hierarchy. *attr_abbrev* is the name attribute abbreviation as defined in the known attribute types file.

WITH "operating_option=state"

Identifies the operating option and its state (on or off) to use for this operation. If WITH "operating_option=state" is not specified, operating options specified when ODSCL was invoked apply; or if operating options are not specified at invocation,default operating parameters apply.

Usage Notes

Attribute information associated with the specified object name is retrieved from the directory and displayed on the terminal.

REGISTER DIRECTORY NAME

REGISTER DIRECTORY NAME — Adds a specified object name and its associated attributes in the directory.

Syntax

```
REGISTER DIRECTORY NAME  "/attr_abbrev=value/attr_abbrev=value . . . "
                         ATTRIBUTES "OC=oc_abbrev/attr_abbrev=value . . . "
                         [WITH "operating_option=state"]
```

Attributes

DIRECTORY

Indicates the entity on which the REGISTER operation is being performed.

NAME "/attr_abbrev=value/attr_abbrev=value . . . "

The quoted attribute name expression that defines the name of the object in the directory hierarchy. *attr_abbrev* is the name attribute abbreviation as defined in the known attribute types file.

ATTRIBUTES "OC=oc_abbrev/attr_abbrev=value . . . "

The set of attributes and the attribute values to be added. oc_abbrev is the object class abbreviation as defined in the known object classes file. *attr_abbrev* is the attribute abbreviation as defined in the known attribute types file. To enter an alias entry, use the attribute expression: "OC=ALIAS/ALIASED_NAME=objectname" for which this is an alias.

WITH "operating_option=state"

Identifies the operating option and its state (on or off) to use for this operation. If WITH "operating_option=state" is not specified, operating options specified when ODSCL was invoked apply; or if operating options are not specified at invocation,the default operating parameters apply.

Usage Notes

The following guidelines apply:

- The object class attribute (attribute abbreviation OC) must be one of the attributes entered.
- If the disable consistency check operating parameter is off (the default state), attributes entered are checked for consistency. The supplied object attributes are checked against the definition of the object class in ods_known_object_classes.dat. The operation fails if an inconsistency exists.

- Attributes defined as mandatory but absent in the command are stored with their default values as specified in ods_known_attribute_types.dat. If no default values exist, the operation fails.
- If the operation is successfully performed, the local directory information is updated with the new information.

Example 3.1, “REGISTER Example” shows the use of the REGISTER command.

Example 3.1. REGISTER Example

```
register directory name "/o-cmps/ou=eng"
attr "oc=ae/cn=eng/p_addr={1 2}"
```

SET DIRECTORY PATH

SET DIRECTORY PATH — Establishes the current name directory path used to refer to the name of an object based on its relative position in the name hierarchy.

Syntax

```
SET DIRECTORY PATH "/attr_abbrev=value/attr_abbrev=value . . . "
```

Attributes

SET

SET is the operation to perform.

DIRECTORY

DIRECTORY is the entity on which the SET command operates.

PATH "/attr_abbrev=value/attr_abbrev=value . . . "

The quoted attribute name expression that defines the name directory path. *attr_abbrev* is the name attribute abbreviation as defined in the known attribute types file.

Usage Notes

The current name directory path is set. Subsequent ODSCL commands can then reference object names using a relative name. For example: "`~/ou=test`"

To set the directory path back to the root level, use the command:

SET DIRECTORY PATH "ROOT"

SHOW DIRECTORY PATH

SHOW DIRECTORY PATH — Displays the current name directory path.

Syntax

```
SHOW DIRECTORY PATH
```

Attributes

SHOW

SHOW is the operation to perform.

DIRECTORY

DIRECTORY is the entity on which the SHOW command operates.

PATH

PATH is the attribute.

Appendix A. Online Help

Table A.1, “Online Help Topic and Subtopic Keywords” lists the valid online HELP topic and subtopic keywords.

Table A.1. Table A.1. Online Help Topic and Subtopic Keywords

HELP Topic	HELP Subtopic
HELP	Invokes the online Help facility.
Local_Directory Commands	Syntax, entering_command_elements, prompts, continuation_lines, displays operating_options
Invoking ODSCL	
EXIT	Leaves the online Help facility.
REGISTER	Syntax, examples, director_prompt, name_prompt, name_expression, attributes_prompt, attributes_expression, with_prompt, with_expression, attribute_abbreviations, operating_options, object_class_abbreviations
DEREGISTER	Syntax, examples, directory_prompt, name_prompt, name_expression, with_prompt, with_expression, attribute_abbreviations, operating_options
READ	Syntax, examples, directory_prompt, name_prompt, name_expression, with_prompt, with_expression, attribute_abbreviations, operating_options
LIST	Syntax, examples, directory_prompt, match_prompt, match_expression, with_prompt, with_expression, to_prompt, file_prompt, attribute_abbreviations, operating_options
MODIFY	Syntax, examples, directory_prompt, name_prompt, name_expression, modify_prompt, add_expression, remove_expression, replace_expression, with_prompt, with_expression, attribute_abbreviations, operating_options
SET	Syntax, directory_prompt, path_prompt, path_expression, examples
SHOW	Syntax, directory_prompt, path_prompt, examples