VMS Software

VSI OpenVMS

OMNI API Omni Definition Facility User Guide

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

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

OMNI API Omni Definition Facility User Guide

Legal Notice

Preface

This document describes the management functions provided to define, monitor, and control selected data in the VSI OMNI Application Program Interface (API) system.

VSI OMNI is an implementation of the Manufacturing Message Specification (MMS) as defined in ISO/IEC standard 9506. The MMS specifies the semantics and syntax for communications between applications running on computer systems and dedicated plant floor processors such as robotic or numeric control (NC) devices or programmable logic controllers (PLCs).

The Omni Definition Facility (ODF) enables a system manager to perform system management and configuration tasks by creating local definitions for remote Virtual Manufacturing Device (VMD) objects.

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 system and network managers and personnel who are experienced in network management or distributed processing.

3. Document Structure

The manual consists of the following chapters and appendices:

Chapter	**Contents**
Chapter 1	describes VSI OMNI ODF user functions.
Chapter 2	provides detailed descriptions of the VSI OMNI ODF commands.
Appendix A	provides a list of the VSI OMNI ODF predefined types.
Appendix B	provides a list of VSI OMNI ODF error messages.
Appendix C	provides a list of supported mappings between MMS and VSI OMNI application types.
Appendix D	provides an example script of object definitions.
Glossary of VSI OMNI Terms	defines the terms used in this document or in reference to the VSI OMNI product set.

4. Related Documents

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

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

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*

5. 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.

6. OpenVMS Documentation

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

7. Typographical Conventions

The following conventions may be 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.
ODF>	The ODF acronym with a right angle bracket is the Omni Definition Facility prompt for user input.
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.
sytem 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.

Convention	Meaning
. . .	Vertical ellipses (dots) in examples indicate that information has been omitted for clarity.

Chapter 1. Using the Omni Definition Facility

The Omni Definition Facility (ODF) enables users to locally create and manage stored definitions and data types for Manufacturing Message Specification (MMS) objects and Virtual Manufacturing Devices (VMDs).

This chapter is organized in the following sections:

- Operations and Functions
- Creating a Virtual Manufacturing Device Definition
- Creating a Domain Definition
- Creating a Program Invocation Definition
- Creating a Variable Definition
- Creating Manufacturing Message Specification and Application Type Definitions
- Committing Definitions to the Database
- Setting the Default Scope
- Deleting a Definition
- Creating, Opening, and Closing a Log File
- Enabling and Disabling Logging
- Displaying Definitions and Current Settings
- Executing Stored Commands
- Creating a Command to Repeat a Definition.

1.1. Operations and Functions

The ODF provides a set of commands and procedures that enable users to perform the following specific operations or functions:

- Create VMD definitions.
- Create MMS Domain definitions and associate them with a locally defined VMD.
- Create MMS Program Invocation (PI) definitions and associate them with a locally defined VMD.
- Create variable definitions and associate them with a locally defined Domain or VMD.
- Create data type definitions.
- Display the local definitions of an MMS object.

- Delete a locally created definition or set of definitions.
- Log the current ODF session to a file for later use.
- Write (export) definition commands for backup or convenience.
- Execute a series of stored commands (for example, commands in a log file).
- Set and display the defaults for an ODF session.

Note

The definitions you create with ODF are local to VSI OMNI but are not necessarily local to the system that is running ODF or using the definitions.

1.1.1. Companion Standards

A companion standard (CS) can function as an integral part of VSI OMNI and can be defined by using ODF.

If a CS exists with VSI OMNI, it can affect the behavior of the VSI OMNI procedure calls. The CS can support objects and attributes that are different from those supported by VSI OMNI.

See your applicable companion standard's guide for details about the objects and attributes supported by the companion standard.

1.1.2. Command Language Interface

The ODF Command Language Interface (CLI) guides you through the correct syntax of each ODF command by supplying prompts and a list of options.

For example, if you want to use the SET command, but cannot remember the exact syntax or choices of the command, enter the SET command without arguments:

```
ODF> SET
(APPLICATION PROFILE, ODF LOGFILE, SCOPE)
_ODF>
```

The CLI prompts you for the next word in the command because the command has been entered in an incomplete form. Options supporting the SET command are listed in parentheses.

1.1.2.1. Level-by-Level Prompting

You can specify the entire command without using CLI, or you can specify part of the command and have CLI prompt only for those words that you miss.

Because CLI displays only supported options, prompting for the options is a good way to check the syntax of a command after receiving a parser error. Any attribute or keyword you specify that is not in the CLI option list is not supported for that command.

1.1.2.2. Command Abbreviations

You can shorten the command line by shortening the number of letters in each word. You can abbreviate any word to the first three characters or to the minimum number of characters that makes the command unique.

1.1.3. Invoking the Facility

You can issue single ODF commands at the Digital Command Language (DCL) prompt or you can invoke the ODF facility for a continuous, interactive session.

To use ODF for single line DCL commands, you must first define the command. For example:

```
$ ODF :== $OMNI_ODF
```

You can then enter single line ODF commands from DCL. For example:

```
$ ODF command option . . .
$ ODF command option . . .
```

Another option is to define ODF as a foreign command:

```
$ ODF :== RUN SYS$SYSTEM:OMNI_ODF
```

or

```
$ ODF = "$SYS$SYSTEM:OMNI_ODF.EXE"
```

You can then use the defined command to invoke an inter- active ODF session:

```
$ ODF
ODF>
```

You can then enter valid ODF commands at the ODF prompt. Each command must be terminated with a semicolon (;) and entered by pressing Return. If you leave out required component ids or attributes, ODF prompts you for them.

1.1.4. Exiting from the Facility

You can enter the EXIT command or press **Ctrl/Z** to exit from ODF and return to the DCL prompt.

The EXIT command attempts a COMMIT before ending the ODF session. If there are unresolved dependencies, ODF does not exit. You must enter additional DEFINE commands to satisfy the dependencies, then reenter the EXIT command.

The QUIT command rolls back any batched DEFINE or DELETE DEFINITION commands and ends the ODF session.

1.1.5. Getting Help

After invoking ODF, you can use the HELP command to display information about individual commands. Enter HELP followed by the name of the command that you want information about:

```
ODF> HELP SET SCOPE
```

This command returns a display of HELP information about the SET SCOPE command. You can also enter the HELP command by itself to display a menu of options.

1.2. Creating a Virtual Manufacturing Device Definition

A complete VSI OMNI VMD definition consists of the items listed in Table 1.1.

Table 1.1. VMD Definitions

Item	Description
vmd_name	The local name of the VMD definition. This name is used to reference the definition; it is not used in communications.
APPLICATION SIMPLE NAME	The name used to look up the application in Omni Directory Services.
CPU	This attribute is not supported for all application profiles. See the appropriate ASE-specific documentation for further information.
DESCRIPTION	Information describing the defined VMD. This is not used in communication.
MAXIMUM SERVICES CALLED	The proposed maximum number of transaction object instances that can be created at the called MMS-user on the association.
MAXIMUM SERVICES CALLING	The proposed maximum number of transaction object instances that can be created at the calling MMS-user on the association.
MAXIMUM SEGMENT SIZE	The proposed maximum size of an MMS message to exchange with the VMD.
MODEL	The proposed maximum size of an MMS message to exchange with the VMD.
NESTING LEVEL	The maximum number of levels of nesting that can occur within any data element over an association with the VMD.
NOT	This attribute is not supported for all application profiles. See the appropriate ASE-specific documentation for further information.
PARAMETER CBB	A list specifying the set of conformance building blocks (CBBs) supported by the VMD.
REVISION	A string describing the software, firmware, or hardware revision level of the VMD.
SUPPORTED SERVICES	A list of services supported by the VMD for the association.
VENDOR	The vendor of the system supporting the VMD.
VERSION	The version of the MMS protocol to use.

To create a VSI OMNI definition of a VMD, enter the DEFINE VMD command, specify the name of the VMD, and supply the values that describe the VMD. For example:

```
ODF> DEFINE VMD myvmd
(APPL,CPU,DESC,MAX,MOD,NEST,NOT,PAR,REV,SUPP,VEN,VERS)
_ODF> APPLICATION SIMPLE NAME mycountry@myorg@myunit@myvmd,
(APPL,CPU,DESC,MAX,MOD,NEST,NOT,PAR,REV,SUPP,VEN,VERS)
_ODF> DESCRIPTION "Example VMD" (',', ';' )
(APPL,CPU,DESC,MAX,MOD,NEST,NOT,PAR,REV,SUPP,VEN,VERS)
_ODF> MAXIMUM SERVICES CALLED 5,
_ODF> MAXIMUM SERVICES CALLING 2,
_ODF> MAXIMUM SEGMENT SIZE 512,
_ODF> MODEL "A",
_ODF> NESTING LEVEL 7,
_ODF> PARAMETER CBB ( NOVALT, NO UNNAMED VARIABLES, TPY ),
_ODF> REVISION "first",
_ODF> SUPPORTED SERVICES
_ODF> VENDOR "me",
_ODF> VERSION 1,
```

```
_ODF> ( NO INFORMATION REPORT,
_ODF> RENAME );
```

To add the definition to the permanent ODF database, enter the COMMIT command (described in Section 1.7.

1.3. Creating a Domain Definition

A complete VSI OMNI definition for a Domain consists of the items listed in Table 1.2.

Table 1.2. Domain Definitions

Item	Description
BLOCK ADDRESS LIST	This attribute is not supported for all application profiles. See the appropriate ASE-specific documentation for more information.
CAPABILITY FILE	An OpenVMS file specifying the capabilities of the Domain.
CONTENT FILE	An OpenVMS file containing the Domain.
vmd_name:domain_ name	The name of the remote Domain object and its associated VMD.
[NO] DELETABLE	A value indicating whether or not the Domain can be deleted from the VMD.
DESCRIPTION	Information describing the defined Domain
[NO] SHARABLE	A value indicating whether or not the Domain can be shared by multiple program invocations.

A Domain definition must include the name of the VMD to which the Domain belongs. ODF rejects any Domain definition that does not specify an existing VMD and capabilities file.

To create a VSI OMNI definition of a Domain, enter the DEFINE DOMAIN command at the ODF prompt and supply the information you need to describe the Domain. For example:

```
ODF> DEFINE DOMAIN myvmd:mydom
(BLOCK ADDRESS LIST, CAPABILITY FILE, CONTENT FILE, [NO]DELETABLE,
DESCRIPTION,
[NO]SHAREABLE)
_ODF> CAPABILITY FILE my_domains:mydom.cap CONTENT FILE
my_domains:mydom.dom
(',', ';' )
_ODF> , DELETABLE, NOSHARABLE;
```

To add the definition to the permanent ODF database, enter the COMMIT command (described in Section 1.7).

1.4. Creating a Program Invocation Definition

A complete VSI OMNI definition of a program invocation (PI) consists of the items listed in Table 1.3.

Table 1.3. Named Variable Definition

Item	Description
vmd_name:pi_name	The name of the program invocation and its associated VMD.

Item	Description
[NO] DELETABLE	A value indicating whether or not the PI can be deleted from the VMD.
[NO] REUSABLE	A value indicating whether or not the PI can be reused.
EXECUTION ARGUMENT STRING	An execution argument that becomes the default for START and RESUME requests for the PI.
monitor_type	One of three values. NO MONITOR indicates that the PI has no monitoring event condition. MONITOR PERMANENT indicates that the PI has a monitoring event condition that exists throughout program execution. MONITOR CURRENT indicates that the PI has a monitoring event condition that exists only for the life of the association.
DOMAIN LIST	A list of references to the Domains that make up this program invocation.
DESCRIPTION	Information describing the defined program invocation.

A PI definition must include the name of the VMD to which the Domain belongs. ODF will reject any PI definition that does not specify an existing VMD.

Each PI definition must also specify a Domain list with at least one Domain in it. ODF will reject the definition if the listed Domains are not defined.

To create a VSI OMNI definition of a program invocation, enter the DEFINE PROGRAM INVOCATION command in response to the ODF prompt and supply the values that describe the PI. For example:

```
ODF> DEFINE PROGRAM INVOCATION myvmd:mypi DOMAIN LIST ( myvmd:mydom )
(',', ';' )
_ODF> , DELETABLE, REUSABLE, XA STRING "/DEBUG", NOMONITOR;
```

To add the definition to the permanent ODF database, enter the COMMIT command (described in Section 1.7).

1.5. Creating a Variable Definition

ODF enables you to create VSI OMNI definitions for the following types of variables:

- Named variables
- Unnamed variables

1.5.1. Named Variables

A complete VSI OMNI definition of a named variable consists of the items listed in Table 1.4.

Table 1.4. Named Variable Definition

Item	Description
vmd_name:domain_ name.variable_name	The name of the remote named variable object and its associated VMD and (optionally) Domain.

Item	Description
type	A reference to a predefined application type. Either a type specification for the variable or a reference to a specification created by the DEFINE APPLICATION NAMED TYPE command.
[NO] DELETABLE	A value that indicates whether the named variable can be deleted from the VMD.
DESCRIPTION	Information that describes the named variable.

A variable definition must include the name of the VMD to which the variable belongs. ODF will reject any definition that does not specify an existing VMD. If a variable is defined as being on a Domain, you must also define the Domain.

You must specify the type of the variable.

To create a VSI OMNI definition for a named variable, enter the DEFINE NAMED VARIABLE command in response to the ODF prompt and supply the required information. For example:

```
DEFINE NAMED VARIABLE Foo:Bar.Y
DESCRIPTION "Domain Bar of VMD Foo Y Coordinate" TYPE INTEGER 32;
DEFINE NAMED VARIABLE Foo:X
DESCRIPTION "VMD Foo X Coordinate" APPLICATION TYPE %:OMNI$LONG;
```

To add the definition to the permanent ODF database, enter the COMMIT command (described in Section 1.7).

1.5.2. Unnamed Variables

A complete VSI OMNI definition of an unnamed variable consists of the items listed in Table 1.5.

Table 1.5. Unnamed Variable Definition

Item	Description
vmd_name:domain_ name.variable_name	The name of the remote unnamed variable and its associated VMD (and, optionally, its Domain).
type	A reference to a predefined application type. Either a type specification for the variable or a reference to a specification created by the DEFINE APPLICATION NAMED TYPE command.
<address>	The address of the unnamed variable.
[NO] Supply Type Spec	A value that indicates whether the variable's type specification is to be sent to the remote VMD to access the variable.
DESCRIPTION	Information that describes the named variable.

To create a VSI OMNI definition for an unnamed variable, enter the DEFINE UNNAMED VARIABLE command in response to the prompt and supply the required information. For example:

```
 ODF> DEFINE UNNAMED VARIABLE myvmd:X APPLICATION TYPE %:OMNI$LONG
 (DES,TYP,APP TYP)
 _ODF> NUMERIC ADDRESS %X4000, DESCRIPTION "Example of an unnamed
 variable";
ODF> DEFINE UV myvmd:AT %:OMNI$LONG SYMBOLIC ADDRESS "$n0:0";
```

To add the definition to the permanent ODF database, enter the COMMIT command (described in Section 1.7).

1.6. Creating Manufacturing Message Specification and Application Type Definitions

An ODF variable definition includes two variable type definitions: an MMS Type definition and an Application Type definition.

- The MMS Type (MT) definition provides information about the variable that is communicated through the MMS protocol when the variable is read or written.
- The Application Type (AT) definition provides information about the way the application views the variable. Application Type information cannot be communicated through the MMS protocol; it is specific to the local programming environment.

ODF provides two commands that you can use to create variable type definitions:

- DEFINE MMS NAMED TYPE. Creates an MMS Type definition.
- DEFINE APPLICATION NAMED TYPE. Creates an Application Type definition and associates the definition with a corresponding MMS Type definition that you have created.

The DEFINE TYPE commands are useful for creating commonly-used type definitions that many variables will reference. When a number of variables refer to the same type definition, all of the variables can be changed by changing the one type definition.

1.6.1. Creating a Manufacturing Message Specification Named Type Definition

A complete VSI OMNI definition of an MMS Named Type consists of the items listed in Table 1.6.

Table 1.6. MMS Named Type Definition

Item	Description
vmd_name:domain_ name.mms_type_name	The name of the MMS Named Type specification and its associated VMD (and, optionally, its Domain).
mms_type_specification	A structure, array or simple type specification or a reference to another MMS Named Type.
[[NO] DELETABLE]	Indicates whether or not the MT can be deleted from the VMD.
DESCRIPTION	Information describing the defined MMS Named Type.

An MMS Named Type definition must include the name of the VMD to which the named type belongs. ODF rejects any definition that does not specify an existing VMD. If a named type is defined as being on a Domain, you must also specify the Domain.

You must specify the MMS type specification.

To create a VSI OMNI definition for an MMS Named Type, enter the DEFINE MMS Named Type command in response to the prompt and supply the required information. For example:

```
 DEFINE MMS NAMED TYPE Foo:Point
 DESCRIPTION "Point in threespace" STRUCTURE
x INTEGER 32;
y INTEGER 32;
z INTEGER 32; END;
```

```
DEFINE MMS NAMED TYPE Foo:Position
DESCRIPTION "Position and Orientation"
STRUCTURE
 pos Foo:Point;
 r FLOAT FORMAT WIDTH 32 EXPONENT 9;
 o FLOAT FORMAT WIDTH 32 EXPONENT 9;
 h FLOAT FORMAT WIDTH 32 EXPONENT 9;
 END;
```

To add the definition to the permanent database, enter the COMMIT command (described in Section 1.7).

1.6.2. Creating an Application Named Type Definition

A complete VSI OMNI definition of an Application Named Type consists of the items listed in Table 1.7.

Table 1.7. Application Named Type Definition

Item	Description
vmd_name:domain_	The name of the Application Named Type specification and its
name.application_type_ name	associated VMD (and, optionally, its Domain).
FROM MMS NAME TYPE	The name of the MMS Named Type associated with the Application Named Type. The default is the same name and scope as the Application Type.
application_type_ specification	A structure, array or simple type specification or a reference to another Application Named Type.
DESCRIPTION	Information describing the defined Application Named Type.

An Application Named Type definition must include the name of the VMD to which the named type belongs. ODF will reject any definition that does not specify an existing VMD. The named type can also be defined on a Domain.

You must specify the application type specification.

To create a VSI OMNI definition for an Application Named Type, enter the DEFINE Application Named Type command in response to the ODF prompt and supply the required information. For example:

```
DEFINE APPLICATION NAMED TYPE Foo:Point
DESCRIPTION "Point in threespace"
APPLICATION TYPE FROM MMS NAMED TYPE Foo:Point STRUCTURE
(x,x) INTEGER 32;
(y,y) INTEGER 32;
(z,z) INTEGER 32; END;
```

To add the definition to the permanent database, enter the COMMIT command (described in Section 1.7).

1.6.3. Creating Application Type Definitions for Alternate Access

Every VSI OMNI variable definition specifies a default Application Type definition, which in turn refers to an MMS Type definition.

Simple applications would generally access the variable's data using the default Application Type. Other applications may need to perform alternate access by referring to the variable using some other Application and/ or MMS Type definition.

One reason for alternate access would be to support applications which store internal data differently. For example, suppose two applications access a variable whose MMS Type definition is a Visible String. One application may need to store this visible string internally as a null terminated string while another Application Type may need to store it internally as a word counted string. In both cases, since all elements of the array would be accessed, there is a 1:1 correspondence between array components of the MMS Type definition and the Application Type definition. Following is an example of the type definitions that can be used under these circumstances:

```
DEFINE MMS NAMED TYPE Foo:String VISIBLE STRING 100; DEFINE APPLICATION
 NAMED TYPE Foo:String
FROM MMS NAMED TYPE Foo:String, STRING 100; DEFINE APPLICATION NAMED TYPE
 Foo:Alt_String_NT
FROM MMS NAMED TYPE Foo:String, NULL TERMINATED STRING 100; DEFINE
 APPLICATION NAMED TYPE Foo:Alt_String_WC
FROM MMS NAMED TYPE Foo:String, WORD COUNTED STRING 100; DEFINE NAMED
 VARIABLE Foo:String_Var APPLICATION TYPE Foo:String;
```

Another reason for alternate access is to support applications which may not need to access all of the data in a variable. This type of alternate access is called partial access. For example, a device can define a portion of its memory as a large array. An application can read the portion of the memory it is interested in by creating an

Application Type definition that specifies a subrange of the array to be read into an application buffer which is only large enough to hold the data in that subrange. Following is an example of the definitions that can be used in circumstances where the default application type references the entire array:

```
DEFINE MMS NAMED TYPE Foo:Int_Array ARRAY [20] of INTEGER 32; DEFINE
 APPLICATION NAMED TYPE Foo:Int_Array
FROM MMS NAMED TYPE Foo:Int_Array, ARRAY [20] of INTEGER 32; DEFINE
 APPLICATION NAMED TYPE Foo:Alt_Int_Array_0_9
FROM MMS NAMED TYPE Foo:Int_Array, ARRAY [0...9] of INTEGER 32;
DEFINE APPLICATION NAMED TYPE Foo:Alt_Int_Array_5_14
FROM MMS NAMED TYPE Foo:Int_Array, ARRAY [5...14] of INTEGER 32;
DEFINE NAMED VARIABLE Foo:Int_Array_Var APPLICATION TYPE Foo:Int_Array;
```

In both examples of alternate access (full and partial), an application accomplishes alternate access on a variable by providing a method handle in calls to the variable access procedures, omni_get_value and omni_put_value.

A method handle is an object identifier handle of an Application Named type.

The user can also define the default application type as an alternate access type. In this case, it is not necessary to supply a method handle to perform alternate access. Instead, alternate access can be performed by default whenever the variable is accessed.

1.7. Committing Definitions to the Database

An ODF session consists of the following steps:

1. The user enters a series of DEFINE and/or DELETE commands to describe the objects in the MMS environment. ODF saves these definitions in a special area allocated for the ODF session.

2. The user enters the COMMIT command. ODF examines the batched definitions (that is, all the definitions entered since the last COMMIT command or since the beginning of the session), writes all valid definitions into the permanent database, or reports on any errors. If committing the changes will produce inconsistencies in the database, referred to as a database constraint error, ODF reports an error and does not make any of the modifications. For example, if you have entered a variable definition that includes a reference to a nonexistent VMD, ODF rejects the definition and returns an error code.

ODF does not discard the batch of definitions if the COMMIT operation fails. Thus, you can correct the error and enter the COMMIT command again.

To erase a batch of modifications from the temporary storage area, enter the ROLLBACK command. ODF discards all the definitions that you have created since your last COMMIT command. (Note how ROLLBACK differs from DELETE. The DELETE command removes a definition that has been committed to the permanent ODF database or exists in temporary storage; the ROLLBACK command discards actions from temporary storage.)

In addition to batching DEFINE commands, ODF batches all commands that modify the database (for example, the DELETE command) until you enter a COMMIT command.

Note

The EXIT command causes ODF to attempt a COMMIT before exiting. The QUIT command causes ODF to attempt a ROLLBACK before exiting.

If, as an ODF user, you arrange your transactions so a COMMIT is issued after each DEFINE command, you can reduce the ambiguity of constraint error messages.

The constraint error message identifiers have the following general format:

DUP<def>	A definition with the name specified in a DEFINE <def> command already exists in the database. To modify a definition, DELETE it, DEFINE it, and then COMMIT it. If modification is not wanted, and the existing database entry is correct, use ROLLBACK to cancel the DEFINE request.
<def1>NO<def2>	An attempt was made to create a definition that is dependent on the existence of another definition. For example, DOMNOVMD means that a DEFINE DOMAIN command was issued for a Domain, and the VMD specified for that Domain does not exist. Either create the required definition or roll back the request. Remember that all definition names are case sensitive.
<def1>REF<def2>	A definition refers to another definition that does not exist. For example, NVREFAT means that a DEFINE NAMED VARIABLE command was issued with an APPLICATION TYPE reference to a nonexisting Application Named Type. Either create the required definition or roll back the request. Remember that all definition names are case sensitive.

Classes of definitions are abbreviated as follows:

VMD: Virtual Manufacturing Device DOM: Domain PI: Program Invocation NV: Named Variable

UV: Unnamed Variable

VAR: Simple variable - named or unnamed (NV or UV) PID: Entry in a PI list of Domains

MT: MMS Named Type

AT: Application Named Type

Appendix B provides a list of the ODF error messages.

1.8. Setting the Default Scope

ODF enables you to set the default VMD and Domain for dependent objects that you want to define. To specify a default VMD and Domain, enter the SET SCOPE command and the name of the VMD and Domain. (If you omit the Domain name, the scope is VMD-specific).

For example, the following SET SCOPE command specifies Foo as the default VMD for the session and Bar as the default Domain. The DEFINE command creates a variable definition named *X*:

```
ODF> SET SCOPE Foo:Bar
ODF> DEFINE NAMED VARIABLE X APPLICATION TYPE %OMNI$LONG;
```

ODF creates the definition Foo:Bar.X.

1.9. Deleting a Definition

The DELETE DEFINITION command deletes a definition from the permanent ODF database and/or temporary storage.

A definition deletion cannot be committed until all of its dependencies are deleted. In the following command sequence, for example, Domain "Bar" cannot be deleted while there is an existing definition for a named variable "Baz" within the "Bar" scope:

```
DEFINE DEFINITION VMD Foo . . .
DEFINE DEFINITION DOMAIN Foo:Bar . . .
DEFINE DEFINITION NAMED VARIABLE Foo:Bar.Baz . . . DELETE Foo:Bar;
COMMIT; ! Will fail because of Foo:Bar.Baz
DELETE DEFINITION Foo:Bar(NAMED VARIABLE:*); ! Delete all variables in
Domain Bar of Vmd Foo
DELETE DEFINITION DOMAIN Foo:Bar;
COMMIT; ! Will succeed DELETE DEFINITION VMD Foo;
```

A right angle bracket character (>) in the command line causes ODF to delete the specified object and all objects that are dependent on that object. For example, the following command deletes VMD Foo and all the objects it contains:

```
DELETE DEFINITION Foo>;
```

To delete the entire database:

```
DELETE DEFINITION *>;
```

Note

VSI recommends that you do not use this command line.

The DELETE command supports the wildcard asterisk (*). For example, the following command deletes all named variables in Domain Foo:Bar:

```
DELETE DEFINITION Foo:Bar(NV:*);
```

1.10. Creating, Opening, and Closing a Log File

ODF enables you to create and open a log file for the ODF session. To create a log file, enter the SET ODF LOGFILE command and specify the name of the log file.

Once the file is open, you can start session logging with the ENABLE ODF LOGGING command. You can also write definition commands to the log file using the WRITE DEFINITION command.

To close the log file, reenter the SET ODF LOGFILE command with a different filename or the null device name (NL:).

1.11. Enabling and Disabling Logging

To create a log of the ODF session, enter the ENABLE ODF LOGGING command.

If you have already specified a log file with the SET ODF LOGFILE command, ODF logs the session to that file. If you have not specified a file, ODF creates a file OMNI_ODF.LOG in the current default directory and logs the session there.

To disable logging, enter the DISABLE ODF LOGGING command. Logging will stop, but the log file will remain open until the ODF session is exited or another SET ODF LOGFILE command is entered.

The ENABLE and DISABLE ODF LOGGING commands can be used to selectively log portions of an ODF command session.

1.12. Displaying Definitions and Current Settings

ODF provides commands that you can use to display the current settings of ODF session attributes or the values of definitions in the database, as shown in Table 1.8.

Table 1.8. ODF Commands

Item	Description
SHOW SCOPE	Shows the default VMD and Domain.
SHOW APPLICATION PROFILE	Shows the current application profile.
SHOW ODF LOGFILE	Shows the current output file.
SHOW ODF LOGGING	Information describing the defined Application Named Type.
SHOW DEFINITION	Displays a definition or set of definitions to SYS $OUTPUT.
SHOW VERSION	Displays the version of ODF and the database level.

1.13. Executing Stored Commands

The DO command (or @) enables you to read ODF commands stored in a script file. You can create script files in any of the following ways:

- Use the SET ODF LOGFILE and ENABLE ODF LOGGING commands to trace a session.
- Use the WRITE DEFINITION TO command to write declaration commands to a file. This is helpful when creating script files to rebuild portions of the database. Issuing a WRITE DEFINITION command without a TO specifier causes a DEFINE command to be written to the current log file.
- Use any editor to create a text file containing the commands.

If logging is enabled when the script file is invoked, the invocation command is commented out in the trace, and the individual commands in the script file appear in the trace output. A comment is inserted at the end of the trace file. If the script file contains an EXIT command, that command does not appear in the trace file.

DO commands can be nested (a script file can issue a DO command). There is no limit to how many DO commands can be issued from a particular script file; however, ODF must open each script file, so the open file limit (FILLM) quota determines the maximum nesting allowed. For example:

The file COMMANDS.COM looks like this:

```
+--------------------------------+
| DEFINE NAMED VARIABLE Bar ... |
| DEFINE NAMED VARIABLE Baz ... |
| EXIT |
+--------------------------------+
$ ODF = "$SYS$SYSTEM:OMNI_ODF.EXE"
$ ODF
ODF> DEFINE VMD Foo; ODF> SET SCOPE Foo; ODF> ENABLE ODF LOGGING
ODF> sho sco Scope is Foo:)
ODF> DEFINE NAMED VAR X ... ODF> DO COMMANDS.COM;
ODF> EXIT
$ TYPE OMNI_DEF.LOG
sho sco
! Scope is Foo:
DEFINE NAMED VAR X ...
! DO COMMANDS.COM;
! Invoking Script File DISK1:[GUEST]COMMANDS.COM;1 DEFINE NAMED VARIABLE
Bar ...
DEFINE NAMED VARIABLE Baz ...
! End of Script File DISK1:[GUEST]COMMANDS.COM;1 EXIT
```

1.14. Creating a Command to Repeat a Definition

The WRITE DEFINITION command enables you to write out definitions to a file, where each definition is written as a valid ODF DEFINE command. A reference to a definition, list of definitions, or a wildcard specification can be specified. An asterisk (*) used as a wildcard character matches zero or more characters, and a period (.) used as a wildcard character matches exactly one character.

If you include a file specification by using the TO clause, ODF opens that file, writes the definitions to it, and closes the file. If there is no file specification, ODF appends the definitions to the current log file. If no log file is open, ODF opens a new version of OMNI_DEF.LOG and writes the definitions there. The name of the file can then be provided in the DO command to execute the commands stored in the file.

The following command writes out all definitions in the database which match the currently set application profile to a file named BACKUP.LOG:

```
WRITE DEFINITION *> TO BACKUP.LOG;
```

To write all definitions for any other application profile, you must first type the SET APPLICATION PROFILE command and specify the wanted application profile. Then enter the WRITE DEFINITION command as shown in the previous example. A new file is created with each WRITE command entered.

The following example writes out Domain Bar of VMD Foo and its dependent objects to file DOMAIN.LOG:

```
WRITE DEFINITION Foo:Bar> TO DOMAIN.LOG;
```

The following example writes named variables defined in Domain Bar to the current log file:

```
WRITE DEFINITION Foo:Bar(NV:*);
```

1.15. Example Script

The ODF command script, OMNI_EXAMPLES:OMNI_ODF_EXAMPLE.COM, provides an example of a command sequence for defining a VMD and its dependent objects then generating a command script of the VMD definition created. Appendix D lists the contents of the example script file.

This script can by run by entering @OMNI_EXAMPLES:OMNI_ODF_EXAMPLE.COM at the ODF prompt.

Chapter 2. Using the Omni Directory Services

This chapter describes the Omni Definition Facility (ODF) commands that create and manage local VSI OMNI definitions of remote Manufacturing Message Specification (MMS) objects.

Table 2.1 lists the conventions used in this chapter.

Table 2.1. Conventions

Symbol	Meaning
[ ]	Square brackets enclose optional expressions.
< >	Angle brackets enclose tokens that must be expanded.
...	Ellipsis (dots) indicate an expression that can be repeated.

A local ODF definition name has the same format as an MMS identifier and is a string of 1 to 32 alphanumeric characters including the dollar sign ($) and underscore (_). However, the identifier cannot begin with a numeric character. Also, ODF definition names, like MMS identifiers, are case-sensitive (for example, foo is not equal to FOO).

Table 2.2 shows the valid formats for referring to a definition with the DELETE DEFINITION, SHOW DEFINITION, and WRITE DEFINITION commands.

Table 2.2. Definition Naming Format and Examples

Definition	Naming Format and Examples
VMD	*vmd_name* vmd
Domain	[*vmd_name* :] *domain_name* vmd:dom, :dom, dom [*vmd_name*] (DOMAIN: *domain_name*) v(DOMAIN:d), (DOM:d)
Program Invocation	[*vmd_name*](PROGRAM INVOCATION: *pi_name*) v(PROGRAM INVOCATION:p), (PI:p)
Named Variable	[*vmd_name* :][*dom_name* .](NAMED VARIABLE: *var_name*) v:d(NAMED VARIABLE:n), :d(NV:n), :(NV:n), (NV:n)
Unnamed Variable	[*vmd_name*](UNNAMED VARIABLE: *uvar_name*) v(UNNAMED VARIABLE:n), (UV:n)
MMS Named Type	[*vmd_name* :][*dom_name* .](MMS NAMED TYPE: *type_name*) v:d(MMS NAMED TYPE:n), :d(MT:n), :(MT:n), (MT:n)
Application Named Type	[*vmd_name* :][*dom_name* .](APPLICATION NAMED TYPE: *type_name*) v:d(APPLICATION NAMED TYPE:n), :d(AT:n), :(AT:n), (AT:n)

2.1. Command Descriptions

COMMIT

COMMIT — Commits changes to the database. All changes made in an ODF session since the last COMMIT become permanent and are visible to other users of the ODF database.

Format

```
COMMIT;
```

Description

When you enter the COMMIT command, ODF processes all of the DEFINE and DELETE DEFINITION commands you have entered since your last COMMIT command or since the beginning of the session.

However, before the modifications are made visible, ODF verifies that those modifications leave the database in a consistent state. If committing the command would cause an inconsistency, ODF reports a constraint violation and the changes are not added to the database. See Appendix B for the list of constraint errors.

To recover from a constraint violation, either roll back the commands or enter additional commands to correct the problem. The SHOW DEFINITION command is useful for pinpointing the cause of the problem. SHOW DEFINITION shows any uncommitted changes as if they had already been applied.

DEFINE APPLICATION NAMED TYPE

DEFINE APPLICATION NAMED TYPE — Creates an Application Named Type definition.

Format

DEFINE APPLICATION NAMED TYPE [<vmd>:] [<dom>] <type> [FROM MMS NAMED TYPE <ref>] <app_type_specification> [DESCRIPTION <text>] ;

Attributes and Values

<vmd>

The name of the VMD to which the Application Named Type belongs. If omitted, ODF uses the default VMD that you have set with the SET SCOPE command.

<vmd> is an MMS identifier.

<dom>

The name of the domain to which the type belongs. If blank, the type's scope is VMD specific. If not specified, ODF uses the default domain that you have set with the SET SCOPE command. (To override the default domain, enter :<named_var_name>.)

<dom> is an MMS identifier.

<type>

The name of the type being defined.

<type> is a character string of up to 255 characters.

FROM MMS NAMED TYPE <ref>

The name of the MMS type to use when using this application type. The MMS type must have the same VMD scope as the application type or must be one of the predefined MMS types. The default is the same name and scope as the application type name.

<ref> is a reference to an MMS named type definition.

<app_type_specification>

A construct specifying local format information. Mapping of an application type specification to an MMS type specification is limited to certain combinations. Structures must be mapped to structures, arrays to arrays, references to references, and simple types to simple types. See Appendix C for details.

The following are valid mapping combinations:

- STRUCTURE

  Indicates that the variable is a structure. A structure entry has the format:

  ```
  STRUCTURE
  {<app_component_id> <app_type_specification>;} ...
  END STRUCTURE
  ```

  <app_component_id> specifies which component of the corresponding MMS structure is being referenced, and what name the application uses to refer to that component. It has the form:

  ```
  <app_component_name>, <mms_component_ name>
  ```

- ARRAY

  Indicates that the variable is an array. The entry has the following format:

  ```
  ARRAY
  <app_array_bounds> OF <app_type_specification>
  ```

  <app_array_bounds> is a positive integer enclosed in brackets (for example, [10]), indicating either the number of elements in the array or the range (for example, [3...5]) indicating which elements in the array are included in a partial access.

  <app_type_specification> indicates that the component type is a structure, array, or simple type.

- BOOLEAN

  Indicates the type is a simple boolean with a cell size of eight bits.

- BIT STRING <cell_size_bits>

  Indicates that the type is a simple bit string. <cell_size_bits> is the number of bits in the bit string. Each bit is stored in the low order bit of an 8-bit cell.

- INTEGER [<cell_size_bits>]

  Indicates that the type is a simple integer.

  <cell_size_bits> is the number of bits to use to represent the integer in two's complement format. Only cell sizes of 8, 16, or 32 are valid. The default is 32.

- UNSIGNED [<cell_size_bits>]

  Indicates that the type is a simple binary integer. <cell_size_bits> is the number of bits to store the value in. Only cell sizes of 8, 16, or 32 are valid. The default is 32.

- F_FLOAT

  Indicates that the type is a simple floating-point, stored locally in VAX F_Float format.

- STRING <size_in_bytes>

  Indicates that the type is a simple scalar byte string. <size_in_bytes> is the length of the string in bytes.

- WORD COUNTED STRING <size_in_bytes>

  Indicates that the type is a word counted string. <size_in_bytes> is the maximum number of characters in the string.

- NULL TERMINATED STRING <size_in_bytes>

  Indicates that the type is a null terminated string. <size_in_bytes> is the maximum number of characters in the string, not including the null terminator.

- OMNITime

  Indicates that the type is a time value stored as six words. It is the time type used internally by the VSI OMNI Application Interface.

- VMS ABSOLUTE TIME

  Indicates that the type is stored as a quadword containing a VMS absolute time value.

- BOOLEAN ARRAY <app_array_bounds>

  Indicates that the type is an array of boolean values, where each value is stored in a cell of eight bits. <app_array_bounds> specifies the number or range of values.

- <ref>

  A reference may be used instead of an explicit type description. The VMD scope of the reference must match the VMD scope of the application named type being defined or must be one of the predefined application named types.

DESCRIPTION <text>

Information identifying the variable.

<text> is a quoted character string with a maximum length of 128 characters.

DEFINE DOMAIN

DEFINE DOMAIN — Creates a definition of a domain and associates the domain with a VMD definition.

Format

DEFINE DOMAIN [<vmd_name>:] <domain_name> [[NO]DELETABLE] [[NO]SHARABLE] [CONTENT FILE <content_filespec>] [CAPABILITY FILE <capability_filespec>] [DESCRIPTION <text>] ;

Attributes and Values

<vmd_name>

The name of the VMD definition with which the domain definition is associated. A <vmd_name> is an MMS identifier.

<domain_name>

The name of the domain.

A <domain_name> is an MMS identifier.

[NO] DELETABLE

Indicates whether or not the domain can be deleted from the VMD. The DELETABLE attribute can be set by a server only. The default is DELETABLE.

[NO] SHARABLE

Indicates whether or not the domain can be shared by multiple program invocations. The default is NO SHARABLE. ODF does not prevent PI definitions from sharing a domain that is marked NO SHARABLE.

CONTENT FILE <content_filespec>

A file containing the domain.

<content_filespec> is an OpenVMS file specification or logical name. The default is "".

CAPABILITY FILE <capability_filespec>

A file specifying the capabilities of the domain.

The <capability_filespec> is an OpenVMS file specification or logical name. The default is OMNI $DOMAINS: [<vmd_name>] <domain_name>.cap.

Note

If the OpenVMS file specification contains a semicolon (;), the specification string must be enclosed in quotes.

DESCRIPTION <text>

Any information identifying the domain. This is not communicated. The default is "".

<text> is a quoted character string with a maximum length of 128 characters.

DEFINE MESSAGE

DEFINE MESSAGE — Creates a local VSI OMNI definition of a message object and associates it with a previously defined VMD. This object is not supported for the application profile. See the appropriate ASE-specific documentation for more information.

Format

DEFINE MESSAGE [<vmd_name>:] <msg_name> LENGTH <msg_length> [DESCRIPTION <text>] ;

Attributes and Values

<vmd_name>

The name of the VMD to which the message belongs. If omitted, ODF uses the default VMD that you have set with the SET SCOPE command.

The <vmd_name> is an MMS identifier.

<msg_name>

The name of the message object being defined. Only one message can be defined for each VMD. The <msg_name> is an MMS identifier.

LENGTH <msg_length>

The maximum length of the message data in bytes.

The <msg_length> is any positive integer in the range from 1 to 4096.

DESCRIPTION <text>

Information identifying the variable.

<text> is a quoted character string with a maximum length of 80 characters. The default is "".

DEFINE MMS NAMED TYPE

DEFINE MMS NAMED TYPE — Creates an MMS Named Type definition. An MMS Named Type definition describes the attributes of a variable that can be communicated to an MMS peer.

Format

DEFINE MMS NAMED TYPE [<vmd>:] [<dom>.] <type> <mms_type_specification> [[NO]DELETABLE] [DESCRIPTION <text>] ;

Attributes and Values

<vmd>

The name of the VMD to which the named type definition belongs. If omitted, ODF uses the default VMD that you have set with the SET SCOPE command.

<vmd> is an MMS identifier.

<dom>

The name of the domain to which the type belongs. If blank, the type's scope is VMD specific. If not specified, ODF uses the default domain that you have set with the SET SCOPE command.

To override the default domain:

:<mms_named_type>.

<dom> is an MMS identifier.

<type>

The name of the MMS type being defined.

<type> is an MMS identifier.

<mms_type_specification>

One of the following values indicating that the variable is a structure, an array, or a simple variable:

- STRUCTURE {<mms_type_component_name> <mms_type_specification>;} ... END;

  Indicates that the value is constructed from an ordered list of one or more components, each of which can have a distinct type.

  <mms_type_component_name> is an MMS identifier.

  <mms_type_specification> describes the type of this component. The type can be a structure, an array, or a simple type.

- ARRAY <mms_array_bounds> OF <mms_type_specification>

  Indicates that the value is an ordered sequence of elements.

  <mms_array_bounds> is a positive integer enclosed in brackets ([ ]).

  <mms_type_specification> describes the type of elements in the array. The type can be a structure, an array, or a simple type.

- BOOLEAN

  Indicates the type is a simple boolean.

- [VARYING] BIT STRING <cell_size_bits>

  Indicates that the type is a simple bit string. <cell_size_bits> is the number of bits in the bit string.

- INTEGER [<cell_size_bits>]

  Indicates that the type is a simple integer. <cell_size_bits> is the number of bits in the largest two's complement number that the integer can hold. The cell size value must be 8, 16, or 32. The default is 32.

- UNSIGNED [<cell_size_bits>]

  Indicates that the type is a simple unsigned integer. <cell_size_bits> is the number of bits in the largest binary number that the unsigned integer can hold. The cell size value must be 8, 16, or 32. The default is 32.

- FLOAT

[FORMAT WIDTH <width>] [EXPONENT <exponent>]

Indicates that the type is a simple floating-point number. The format width is 32 bits and the exponent is 8 bits.

<width> is the largest width in bits. <exponent> is the exponent width in bits.

- [VARYING] OCTET STRING <size_in_octets>

  Indicates that the type is a simple octet string. Each octet can hold a value from 0 to 255. <size_in_octets> is the number of octets in the string.

- [VARYING] VISIBLE STRING <size_in_octets>

  Indicates that the type is a simple visible string. <size_in_octets> is the number of characters in the string. The string should hold a printable ASCII value.

- GENERALIZED TIME

  Indicates that the type is a generalized time value.

- BINARY TIME DATE [NOT] INCLUDED

  Indicates that the type is a binary time value. DATE INCLUDED is the default.

- BCD [<size_in_digits>]

  Indicates that the type is an unsigned binary coded decimal number.

  <size_in_digits> is the number of decimal digits used to represent the maximum value that the variable can hold.

- <ref>

  A reference to another MMS Named Type can be used instead of an explicit type description. The VMD scope of the reference must match the VMD scope of the MMS Named Type being defined or must be one of the predefined MMS Named Types.

[NO] DELETABLE

Indicates whether the MMS Named Type can be deleted from the VMD. The default is DELETABLE.

DESCRIPTION <text>

Information identifying the type.

<text> is a quoted character string with a maximum length of 128 characters. The default is "".

DEFINE NAMED VARIABLE

DEFINE NAMED VARIABLE — Creates a VSI OMNI definition of a named variable and associates the definition with a defined domain or VMD.

Format

DEFINE NAMED VARIABLE [<vmd>:] [<dom>.] <var> <type> [[NO]DELETABLE] [DESCRIPTION <text>] ;

Attributes and Values

<vmd>

The name of the VMD to which the variable belongs. If omitted, ODF uses the default VMD that you have set with the SET SCOPE command.

<vmd> is an MMS identifier.

<dom>

The name of the domain to which the variable belongs. If not specified, ODF uses the default scope that you have set with the SET SCOPE command. See the SET SCOPE command for details.

<dom> is an MMS identifier.

<var>

The name of the named variable being defined.

<var> is an MMS identifier.

<type>

There is one variable type:

APPLICATION TYPE <app_type_reference>

Where:

<app_type_reference> is a reference to a predefined application type, such as %:OMNI$LONG, or to a user- defined application type. (See DEFINE APPLICATION NAMED TYPE.)

You must enter a type. However, the type you specify can be overridden at run time by means of a method handle. See Section 1.6.3 for more information. See Appendix A for a list of the available predefined types.

[NO] DELETABLE

Indicates whether the variable can be deleted from the VMD. The default is DELETABLE.

DESCRIPTION <text>

Information identifying the variable.

<text> is a quoted character string with a maximum length of 128 characters. The default is " ".

DEFINE PROGRAM INVOCATION

DEFINE PROGRAM INVOCATION — Creates a definition of a Program Invocation (PI) and associates the PI with a VMD definition.

Format

DEFINE PROG INVOC [<vmd_name>:] <pi_name> [DESCRIPTION <text>] DOMAIN LIST <dom_id>[,<dom_id>] ... [[NO]DELETABLE] [[NO]REUSABLE] [[NO]EXECUTION ARG STRING <text>] [[NO]MONITOR <type>] ;

Attributes and Values

<vmd_name>

The name of the VMD definition with which the domain definition is associated. The <vmd_name> is an MMS identifier.

<pi_name>

The name of the program invocation. The <pi_name> is an MMS identifier. **DESCRIPTION <text>**

Any information identifying the PI.

<text> is a quoted character string with a maximum length of 128 characters. The default is " ".

DOMAIN LIST <dom_id>[,<dom_id>] ...

A list of one or more domain references.

<dom_id> is an MMS identifier. Separate the IDs with commas and enclose the list in parentheses. At least one domain must be specified. The order of the list is not significant.

[NO] DELETABLE

Indicates whether or not the program invocation can be deleted from the VMD. The default is DELETABLE.

[NO] REUSABLE

An execution argument for the program invocation. If supplied, the value becomes the default for both START and RESUME service requests for the program invocation. The value can be overridden on the call to either the OMNI START or RESUME function. The default is NOEXECUTION ARGUMENTSTRING.

<text> is a symbol or a quoted text string. The maximum length is 128 characters.

[NO] MONITOR <type>

Indicates whether program monitoring is in effect. NO MONITOR is the default. If you specify MONITOR, one of the following monitoring types must be entered:

- MONITOR CURRENT — Indicates that the PI has a monitoring event condition that exists for the life of the association.
- MONITOR PERMANENT — Indicates that the PI has a monitoring event condition that exists throughout program execution.

DEFINE UNNAMED VARIABLE

DEFINE UNNAMED VARIABLE — Creates a VSI OMNI definition of an unnamed variable object and associates the definition with a defined domain or VMD.

Format

DEFINE UNNAMED VARIABLE [<vmd>:] <var> <type> <address> [[NO]Supply Type Spec] [DESCRIPTION <text>] ;

Attributes and Values

<vmd>

The name of the VMD to which the variable belongs. If omitted, ODF uses the default VMD that you have set with the SET SCOPE command.

<vmd> is an MMS identifier.

<var>

The name of the unnamed variable being defined.

<var> is an MMS identifier.

<type>

There is one variable type:

APPLICATION TYPE <app_type_reference>

Where:

<app_type_reference> is a reference to a predefined application type, such as %:OMNI$LONG, or to a user- defined application type. (See DEFINE APPLICATION NAMED TYPE.

You must enter a type. However, the type you specify can be overridden at run time by means of a method handle. See Section 1.6.3 for more information. See Appendix A for a list of the available predefined types.

<address>

Use one of the following to indicate the address of the variable:

- NUMERIC ADDRESS <longword_value>

  <longword_value> can be a decimal number (the default) or hexadecimal number. A hexadecimal number has the format %hhhhhhhh where each h is a hexadecimal digit (0–9, a–f, or A–F).

- SYMBOLIC ADDRESS <address_string>

  <address string> should be a string enclosed in double quotation marks (" ").

[NO] Supply Type Spec

Indicates whether the variable's type description is to be sent to the remote VMD with requests to access this variable. The default is NOSupply Type Spec.

DESCRIPTION <text>

Information identifying the variable.

<text> is a quoted character string with a maximum length of 128 characters. The default is " ".

DEFINE VMD

DEFINE VMD — Creates a local VSI OMNI definition of a VMD.

Format

DEFINE VMD <vmd_name> APPLICATION SIMPLE NAME <app_simple_name> [VERSION <version_number>] [NESTING LEVEL <word_value>] [MAXIMUM SERVICES CALLED <word_value>] [MAXIMUM SERVICES CALLING <word_value>] [MAXIMUM SEGMENT SIZE <integer_value>] [PARAMETER CBB <cbb_list>] [SUPPORTED SERVICES <supported_service_list>] [[NO]VENDOR <vendor_name>] [[NO]MODEL <model>] [[NO]REVISION <revision>] [DESCRIPTION <text>] ;

Attributes and Values

<vmd_name>

The local name of the VMD definition.

The <vmd_name> is an MMS identifier. The VMD name is used to identify the VMD in the Omni database; it is not used for communications.

APPLICATION SIMPLE NAME <app_simple_name>

The simple name is used to look up the application in Directory Services. The format of the name is determined by the Directory Service Provider you are using. See the *VSI OMNI API Omni Directory Services User Guide* for further details. If you omit the simple name, VSI OMNI uses the VMD name.

When the Omni Directory Services Command Language (ODSCL) is used as the directory services provider, the Application Simple Name is normally set to the Common Name of the desired ODS entry. The Application Simple Name must include the prefix /cn= if an MMS Application Profile is being used.

The <app_simple_name> is a quoted character string of up to 128 characters.

VERSION <version_number>

Specifies the protocol used by the application.

The <version_number> is an integer value. The range is determined by the application profile being used. For the MMS companion standard, the following values are valid:

- Zero (0) — The application is DIS compliant or is compliant with the draft international standard of ISO/ISE standard 9506.
- One (1) — The application is IS compliant or is compliant with ISO/ISE standard 9506. This is the default.

NESTING LEVEL <word_value>

The maximum number of levels of nesting that can occur within any data element that is transmitted or communicated over an association with the VMD. A value of zero (0) specifies unlimited nesting. The default is 10.

The <LEvel> is an integer.

MAXIMUM SERVICES CALLING <word_value>

The proposed maximum number of transaction object instances that can be created at the calling MMS-user on the application association. The default is 5.

The <word_value> is an integer.

MAXIMUM SERVICES CALLED <word_value>

The proposed maximum number of transaction object instances that can be created at the called MMS-user on the application association. The default is 5.

The <word_value> is an integer.

MAXIMUM SEGMENT SIZE <integer_value>

The proposed maximum size of an MMS message to exchange with a VMD. The default size is 512. A value between 64 and 8192 is allowed.

PARAMETER CBB <cbb_list>

The set of conformance building blocks supported by the VMD.

The <cbb_list> consists of one or more of the items listed in Table 2.3, separated by commas and enclosed in parentheses.

Table 2.3. CBB Parameters

Parameter	ISO/ISE 9506 Designation
[NO] ARRAYS	STR1
[NO] STRUCTURES	STR2
[NO] NAMED VARIABLES	VNAM
[NO] ALTERNATE ACCESS	VALT
[NO] UNNAMED VARIABLES	VADR
[NO] SCATTERED ACCESS	VSCA
[NO] THIRD PARTY	TPY
[NO] NAMED VARIABLE LIST	VLIS
[NO] REAL	REAL
[NO] ACKNOWLEDGEMENT EVENT CONDITION	AKEC

Parameter	ISO/ISE 9506 Designation
[NO] EVALUATION INTERVAL	CEI

SUPPORTED SERVICES <supported_service_list>

The set of services supported by the calling MMS user for the association.

The <supported_service_list> consists of one or more of the services listed in Table 2.4, separated by commas and enclosed in parentheses.

Table 2.4. Supported Services

Services	
[NO] STATUS	[NO] RESUME
[NO] GET NAME LIST	[NO] RESET
[NO] IDENTIFY	[NO] KILL
[NO] RENAME	[NO] GET PROGRAM INVOCATION ATTRIBUTES
[NO] READ	[NO] OBTAIN FILE
[NO] WRITE	[NO] DEFINE EVENT
[NO] GET VARIABLE ACCESS ATTRIBUTES	[NO] DELETE EVENT CONDITION
[NO] DEFINE NAMED VARIABLE	[NO] GET EVENT CONDITION ATTRIBUTES
[NO] DEFINE SCATTERED ACCESS	[NO] REPORT EVENT CONDITION STATUS
[NO] GET SCATTERED ACCESS ATTRIBUTES	[NO] ALTER EVENT CONDITION MONITORING
[NO] DELETE VARIABLE ACCESS	[NO] TRIGGER EVENT
[NO] DEFINE NAMED VARIABLE LIST	[NO] DEFINE EVENT ACTION
[NO] GET NAMED VARIABLE LIST ATTRIBUTES	[NO] DELETE EVENT ACTION
[NO] DELETE NAMED VARIABLE LIST	[NO] GET EVENT ACTION ATTRIBUTES
[NO] DEFINE NAMED TYPE	[NO] REPORT EVENT ACTION STATUS
[NO] GET NAMED TYPE ATTRIBUTES	[NO] DEFINE EVENT ENROLLMENT
[NO] DELETE NAMED TYPE	[NO] DELETE EVENT ENROLLMENT
[NO] INPUT	[NO] ALTER EVENT ENROLLMENT
[NO] OUTPUT	[NO] REPORT EVENT ENROLLMENT STATUS
[NO] TAKE CONTROL	[NO] GET EVENT ENROLLMENT ATTRIBUTES
[NO] RELINQUISH CONTROL	[NO] ACKNOWLEDGE EVENT NOTIFICATION
[NO] DEFINE SEMAPHORE	[NO] GET ALARM SUMMARY
[NO] DELETE SEMAPHORE	[NO] GET ALARM ENROLLMENT SUMMARY

Services	
[NO] REPORT SEMAPHORE STATUS	[NO] READ JOURNAL
[NO] REPORT POOL SEMAPHORE STATUS	[NO] WRITE JOURNAL
[NO] REPORT SEMAPHORE ENTRY STATUS	[NO] INITIALIZE JOURNAL
[NO] INITIATE DOWNLOAD SEQUENCE	[NO] REPORT JOURNAL STATUS
[NO] DOWNLOAD SEGMENT	[NO] CREATE JOURNAL
[NO] TERMINATE DOWNLOAD SEQUENCE	[NO] DELETE JOURNAL
[NO] INITIATE UPLOAD SEQUENCE	[NO] GET CAPABILITY LIST
[NO] UPLOAD SEGMENT	[NO] FILE OPEN
[NO] TERMINATE UPLOAD SEQUENCE	[NO] FILE READ
[NO] REQUEST DOMAIN DOWNLOAD	[NO] FILE CLOSE
[NO] REQUEST DOMAIN UPLOAD	[NO] FILE RENAME
[NO] LOAD DOMAIN CONTENT	[NO] FILE DELETE
[NO] STORE DOMAIN CONTENT	[NO] FILE DIRECTORY
[NO] DELETE DOMAIN	[NO] UNSOLICITED STATUS
[NO] GET DOMAIN ATTRIBUTES	[NO] INFORMATION REPORT
[NO] CREATE PROGRAM INVOCATION	[NO] EVENT NOTIFICATION
[NO] DELETE PROGRAM INVOCATION	[NO] ATTACH TO EVENT CONDITION
[NO] START	[NO] ATTACH TO SEMAPHORE
[NO] STOP	[NO] CONCLUDE
[NO] CANCEL	

[NO] VENDOR <vendor_name>

The name of the vendor of the system that supports this VMD, enclosed in double quotes (" "). The default is NOVENDOR. VSI OMNI uses the default vendor name. The vendor name is relevant only when you are defining a server VMD.

The <vendor_name> is a character string with a maximum length of 128 characters.

[NO] MODEL <model>

The model of the system supported by the VMD, enclosed in double quotes (" ").

The default is NO MODEL. VSI OMNI uses the default model name. The model name is relevant only when you are defining a server VMD.

<model> is a character string with a maximum length of 128 characters.

[NO] REVISION <revision>

The name of the revision of the system that supports this VMD, enclosed in double quotes (" "). The default is NOREVISION. VSI OMNI uses the default revision. The revision is relevant only when you are defining a server VMD.

<revision> is a character string with a maximum length of 128 characters.

DESCRIPTION <text>

Any information identifying the VMD.

<text> is a quoted character string with a maximum length of 128 characters.

DELETE DEFINITION

DELETE DEFINITION, — Removes a definition from the database. A definition cannot be deleted until all of its dependent definitions have been deleted.

Format

DELETE DEFINITION <def_ref> [, <def_ref>, ...] ;

Attributes and Values

<def_ref>

The reference to a definition. See Table 2.2 for the form of a reference.

The DELETE DEFINITION command supports the following special characters in definition references as shown in Table 2.5.

Table 2.5. DELETE DEFINITION Special Characters

Character	Meaning
*	Asterisk wildcard. Matches zero (0) or more characters in a name. For example, "a*z" matches "az", "abz", "abcz", and so forth.
.	Period wildcard. Matches exactly one character in a name. For example, "a.z" matches "abz", but not "az" or "abcz".
def_ref>	Definition reference followed by a right angle bracket. Deletes the definition and all definitions that are dependent on the definition. Can only be used with a VMD or Domain reference.

DISABLE ODF LOGGING

DISABLE ODF LOGGING — Stops logging of the current ODF session. The logfile is not closed. To close the logfile, issue a SET ODF LOGFILE command or exit from the session.

Format

DISABLE ODF LOGGING;

DO

DO — Executes a series of stored commands, such as those saved in a script file by the ENABLE command. DO is a synonym for the "at" symbol (@).

Format

DO <script_file>;

Attributes and Values

<script_file>

An OpenVMS file specification or logical name pointing to the script file. The default file extension for a script file is .COM.

ENABLE ODF LOGGING

ENABLE ODF LOGGING — Enables logging to the logfile specified in the most recent SET ODF LOGFILE command. If no logfile has been set since the start of the ODF session, ODF tries to create the file OMNI_ODF.LOG in the current default directory and log commands to that file.

Format

```
ENABLE ODF LOGGING;
```

EXIT

EXIT — Commits any outstanding changes to the database and exits from ODF. If the outstanding changes are invalid, ODF reports an error and does not exit.

Format

```
EXIT;
```

QUIT

QUIT — Cancels all of the DEFINE and DELETE commands you have entered since the last COMMIT command, then exits from the session. No definition data from the cancelled commands is written to the database.

Format

```
QUIT;
```

ROLLBACK

ROLLBACK — Cancels all of the DEFINE and DELETE DEFINITION commands you have entered since the last COMMIT command. No definition data is written to the database from commands within the range of the rollback.

Format

```
ROLLBACK;
```

SET APPLICATION PROFILE

SET APPLICATION PROFILE — Sets the application profile for subsequent DEFINE and SHOW operations. A VMD and its dependencies must all be defined under the same application profile.

Format

SET APPLICATION PROFILE <application_profile_name>;

Description

Definitions created after the application profile is set are associated with the Application Service Element (ASE) and the Companion Standard (CS) associated with the application profile. Setting the application profile implicitly identifies the abstract syntax used. See the enumeration omni_l_app_profile of the omni_defs file for more information.

Attributes and Values

<application_profile_name>

The following application profile name:

MMS (default)

For MMS, Application Service Element is the MMS International Standard and no Companion Standard is used.

OSAP

See the OSAP specification for more information.

SET ODF LOGFILE

SET ODF LOGFILE — Specifies the file to which ODF logs the session.

Format

SET ODF LOGFILE <vms_file_specification>;

Description

The log file is opened immediately, but logging is disabled until you enter an ENABLE ODF LOGGING command. If a log file is already open, ODF closes the file before attempting to open the new file. If logging is currently enabled, ODF issues an implicit DISABLE ODF LOGGING command.

Attributes and Values

<vms_file_specification>

An OpenVMS file specification for a file to receive the session log. The default specification is OMNI_ODF.LOG.

SET SCOPE

SET SCOPE — The scope of an ODF command is the VMD or domain definition, or both, that the command is associated with. The SET SCOPE command specifies the definitions that ODF uses as the default scope.

Format

SET SCOPE [<vmd_name>] [:<domain_name>] ;

Attributes and Values

<vmd_name>

The VMD to use as the default.

<domain_name>

The domain to use as the default. If you leave the domain name blank, the default scope is VMD-specific. To change to a different domain in the same VMD, enter the command:

SET SCOPE :<domain_name>;

Where <domain_name> is the name of the new domain. To blank out the scope setting, enter the command:

```
SET SCOPE;
```

The following examples provide Named Variable definitions with scopes set.

```
SET SCOPE v;   ! default scope is now set to VMD "v";
DEF NV x ...   ! defines a variable on VMD "v"; equivalent to the command "DEF NV v:x..."
DEF NV a.y ... ! defines a variable on domain "a" of VMD "v"; equivalent to the command "DEF
 NV v:a.y..."
DEF NV w:z ... ! overrides the default VMD scope and defines a variable on VMD "w";
 equivalent to the command "DEF NV w:z..."
SET SCOPE v:a; ! default scope is now set to VMD "v", domain "a"
DEF NV x ...   ! defines a variable on domain "a"; equivalent to "DEF NV v:a.x...";
DEF NV b.x ... ! overrides the default domain scope and defines a variable on domain "b";
 equivalent to the command "DEF NV v:b.x..."
DEF NV :x ...  ! overrides the default domain scope and defines a variable on VMD "v";
 equivalent to the command "DEF NV v:x..."
```

SHOW

SHOW — Displays current ODF session settings.

Format

SHOW <setting>;

Attributes and Values

<setting>

One of the following settings for the ODF session:

- APPLICATION PROFILE
- DEFINITION
- ODF LOGFILE
- ODF LOGGING
- SCOPE

- VERSION

SHOW DEFINITION

SHOW DEFINITION — Displays definitions from the database. If modifications have been made, but not committed, they are also visible.

Format

SHOW DEFINITION <def_ref> [, <def_ref>, ...] ;

Attributes and Values

<def_ref>

The reference to a definition. See Table 2.2 for the form of a reference.

The SHOW DEFINITION command supports special characters in definition references as shown in Table 2.6.

Table 2.6. SHOW DEFINITION Special Characters

Character	Meaning
*	Asterisk wildcard. Matches zero (0) or more characters in a name. For example, "a*z" matches "az", "abz", "abcz", and so forth.
.	Period wildcard. Matches exactly one character in a name. For example, "a.z" matches "abz", but not "az" or "abcz"
def_ref>	Definition reference followed by a right angle bracket. Displays the definition and all definitions that are dependent on the definition. Can only be used with a VMD or Domain reference.

WRITE DEFINITION

WRITE DEFINITION — Writes out definitions to a file. Each definition is written as a valid ODF command.

Format

WRITE DEFINITION <def_ref> [, <def_ref>, ...] [TO <filespec>] ;

Description

If you include a file specification, ODF opens the file, writes the definitions, and closes the file. If you omit the file specification, ODF appends the definitions to the current log file (if there is no open log file, ODF opens a new version of OMNI_ODF.LOG and writes the definitions).

Attributes and Values

<def_ref>

A reference to a definition or set of definitions. If you enter multiple references, separate them with commas. See Table 2.7 for examples of definition references.

<filespec>

An OpenVMS file specification for a file to contain the definition. If not specified, the command is written to the log file.

The WRITE DEFINITION command supports special characters in definition references as shown in Table 2.7.

Table 2.7. WRITE DEFINITION Special Characters

Character	Meaning
*	Asterisk wildcard. Matches zero (0) or more characters in a name. For example, "a*z" matches "az", "abz", "abcz", and so forth.
.	Period wildcard. Matches exactly one character in a name. For example, "a.z" matches "abz", but not "az" or "abcz".
def_ref>	Definition reference followed by a right angle bracket. Writes the definition and all definitions that are dependent on the definition. Can only be used with a VMD or Domain reference.

Appendix A. Predefined Types

Table A.1 lists the predefined types supported by the Omni Definition Facility (ODF).

Table A.1. Predefined Types

Predefined Type	Meaning
OMNI$BIT8	8-bit bitstring transmitted as bitstring
OMNI$BIT16	16-bit bitstring transmitted as bitstring
OMNI$WC_STR4	4-byte word-counted string, transmitted as varying octet string
OMNI$WC_STR8	8-byte word-counted string, transmitted as varying octet string
OMNI$WC_STR10	10-byte word-counted string, transmitted as varying octet string
OMNI$WC_STR16	16-byte word-counted string, transmitted as varying octet string
OMNI$WC_STR18	18-byte word-counted string, transmitted as varying octet string
OMNI$WC_STR32	32-byte word-counted string, transmitted as varying octet string
OMNI$WC_FIXED_STR4	4-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR6	6-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR8	8-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR10	10-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR16	16-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR18	18-byte word-counted string, transmitted as fixed octet string
OMNI$WC_FIXED_STR32	32-byte word-counted string, transmitted as fixed octet string
OMNI$LONG	32-bit signed integer
OMNI$WORD	16-bit signed integer
OMNI$BYTE	8-bit signed integer
OMNI$ULONG	32-bit unsigned integer
OMNI$UWORD	16-bit unsigned integer

Predefined Type	Meaning
OMNI$UBYTE	8-bit unsigned integer
OMNI$BOOLEAN	8-bit boolean, transmitted as boolean
OMNI$BIT32	32-bit bitstring, transmitted as bitstring
OMNI$F_FLOAT	F_FLOATING transmitted as FLOAT
OMNI$NT_STR4	4-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR6	6-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR8	8-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR10	10-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR16	16-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR18	18-byte null-terminated string, transmitted as varying visible string
OMNI$NT_STR32	32-byte null-terminated string, transmitted as varying visible string
OMNI$NT_FIXED_STR4	4-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR6	6-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR8	8-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR10	10-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR16	16-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR18	18-byte null-terminated string, transmitted as a fixed visible string
OMNI$NT_FIXED_STR32	32-byte null-terminated string, transmitted as a fixed visible string

Appendix B. Error Messages

This appendix lists the Omni Definition Facility (ODF) error messages.

ATCDUPNAME

Application Type structure contains duplicate component names.

ATCNOATS

Application Type Component depends on nonexistent Application Type Specification.

ATCNOMTC

Cannot resolve Application Type structure component reference to MMS Named Type Component.

ATCREFATS

Application Type Component refers to nonexistent Application Type Specification.

ATCREFMISS

Cannot resolve Application Type structure component reference to MMS Named Type.

ATCREFMTC

Application Type Component refers to nonexistent MMS Type Component.

ATCSTRORD

Ordering of Application Type structure components does not match MMS Named Type.

ATNODOM

The Domain that an Application Named Type was defined on does not exist.

ATNOMT

The MMS Named Type referred to in the FROM clause of an Application Named Type definition does not exist.

ATNOVMD

The VMD that an Application Named Type was defined on does not exist.

ATSMAPMTS

Mapping of an Application Type to MMS Named Type is not supported.

ATSNOAT

Application Named Type depends on nonexistent Application Named Type.

ATSREFINV

Type of Application Type reference does not match type of MMS Named Type reference.

ATSREFMTS

Cannot resolve Application Type reference to MMS Named Type.

DOMNOVMD

The VMD that a Domain was defined for does not exist.

DUPAPP

The VMD that a Domain was defined for does not exist.

DUPAPP

Duplicate Application (OSAP only).

DUPAT

Duplicate Application Named Type.

DUPATC

Duplicate Application Type Component.

DUPATS

Duplicate Application Type Specification.

DUPDOM

Duplicate Domain Definition.

DUPMT

Duplicate MMS Named Type.

DUPMTC

Duplicate MMS Type Component.

DUPMTS

Duplicate MMS Type Specification.

DUPNV

Duplicate Named Variable.

DUPPI

Duplicate Program Invocation Definition.

DUPPID

Duplicate entry in a Program Invocation list of Domains.

DUPUDA

(Internal.)

DUPUDC

(Internal.)

DUPUDF

(Internal.)

DUPUV

Duplicate Unnamed_Variable.

DUPVLE

Duplicate Variable List entry.

DUPVLS

Duplicate Variable List.

DUPVMD

Duplicate VMD Definition.

DUPVRS

(Internal.)

MSGNOVMD

(Internal.)

MTCNOMTS

MMS Type component depends on nonexistent MMS Type Specification.

MTCREFMTS

MMS Type Component refers to nonexistent MMS Type Specification.

MTNODOM

The Domain an MMS Named Type was defined on does not exist.

MTNOVMD

The VMD that an MMS Named Type was defined on does not exist.

MTSNOMT

MMS Type specification depends on nonexistent MMS Named Type.

NVNODOM

The Domain that a Named Variable was defined on does not exist.

NVNOVMD

The VMD that a Named Variable was defined on does not exist.

NVREFAT

A Named Variable definition refers to an Application Named Type that does not exist.

ONEMSGVMD

(Internal.)

PIDNOPI

(Internal.)

PINOVMD

The VMD that a program invocation was defined on does not exist.

PIREFDOM

One or more Domains listed in a PI Domain List are not defined.

UVNODOM

The Domain in which an Unnamed_Variable was defined does not exist.

UVNOVMD

The VMD in which an Unnamed_Variable was defined does not exist.

UVREFAT

An Unnamed_Variable definition refers to an Application Named Type that does not exist.

VLENOVLS

(Internal.)

VLSNODOM

(Internal.)

VLSNOVMD

(Internal.)

VLSREFAT

(Internal.)

VLSREFVAR

(Internal.)

Appendix C. Supported Mappings

Table C.1 lists supported mappings between MMS and Application Types.

Table C.1. Supported Mappings

MMS Type	Application Type
BOOLEAN	BOOLEAN
BOOLEAN	INTEGER 8
INTEGER n, n <= 8	INTEGER 8
INTEGER n, n <= 16	INTEGER 16
INTEGER n, n <= 32	INTEGER 32 (default)
UNSIGNED n, n <= 8	UNSIGNED 8
UNSIGNED n, n <=16	UNSIGNED 16
FLOAT (exponent 8, format 32)	F_FLOAT
BIT STRING n	BIT STRING x, x = n
BIT STRING n	BOOLEAN ARRAY x, x = n
[VARYING] BIT STRING n	WORD COUNTED STRING x, x >=n, x<=65535
OCTET STRING n	STRING x, x = n
[VARYING] OCTET STRING n	WORD COUNTED STRING x, x >= n, x<=65535
VISIBLE STRING n	STRING x, x = n
[VARYING] VISIBLE STRING n	NULL TERMINATED STRING x, x >= n
[VARYING] VISIBLE STRING n	WORD COUNTED STRING x, x >= n, x<=65535
GENERALIZED TIME	VMS ABSOLUTE TIME
BINARY TIME DATE INCLUDED	VMS ABSOLUTE TIME
BINARY TIME DATE NOT INCLUDED	VMS ABSOLUTE TIME
BCD n, n <= 8	UNSIGNED 32
OBJECT IDENTIFIER	STRING n
OBJECT IDENTIFIER	WORD COUNTED STRING n, n <=65535
OBJECT IDENTIFIER	NULL TERMINATED STRING n
ARRAY [n] OF <MMS type x>	ARRAY [s] OF <application type y> where s <= n and x and y are a supported mapping
ARRAY [n] OF <MMS type x>	ARRAY [s1..s2] OF <application type y> where s1 <= n, s2 <= n, s1 <= s2, and x and y are a supported mapping
STRUCTURE	STRUCTURE

Appendix D. Example Script of Object Definitions

The following is an example script of object definitions that defines a Virtual Manufacturing Device (VMD) and its dependent objects, commits the definitions, creates an Omni Definition Facility (ODF) command file to redefine the VMD, then executes the command file.

```
! delete example_vmd and all dependent objects
DELETE DEFINITION example_vmd>;
! define a new VMD
DEFINE VMD example_vmd APPLICATION SIMPLE NAME /cn=OMNI_IVP_INIT_VMD,
 DESCRIPTION "Example VMD", VERSION 1, NEST LEVEL 5, MAX SERV CALLING 10,
MAX SERV CALLED 10, MAX SEG 1024, MODEL "Model Name", VENDOR "Vendor Name",
REVISION "V X.X",
PARAM CBB (ARRAYS, STRUC, NAMED VAR, ALT ACC, UNNAMED VAR),
SUPP SER (READ, WRITE, STAT, START, STOP, NO DELE DOM, IDENTIFY, NORENAME);
! set the scope to example_vmd
! all object definitions which follow will be associated with this
! VMD scope unless the object name includes an explicit scope, SET SCOPE
 example_vmd;
! define domains on example_vmd
DEFINE DOMAIN example_vmd:Dom1 CONTENT FILE domain1.content CAPABILITY FILE
 Domain1.cap DELETABLE, NO SHARABLE;
DEF DOM Dom2 NO DEL SHAR;
! define program invocations on example_vmd
DEF PROGRAM INVOCATION example_vmd:PI1 DOMAIN LIST ( Dom1, Dom2),
 DELETABLE, NO REUSABLE, MONITOR CURRENT, NO EXEC ARG;
DEF PI PI2 DOM LIS (Dom2) NO DEL, REU, MON PERM, EXEC ARG STRING "start";
! define simple mms named types on example_vmd DEFINE MMS NAMED TYPE
 example_vmd:BCD BCD 8; DEF MT BinTimeDate BINARY TIME DATE;
DEF MT BinTimeNoDate BIN DATE NOT INC; DEF MT BitStr BIT STRING 32;
DEF MT BitStrVar VAR BIT STR 32; DEF MT Boolean BOOLEAN;
DEF MT Float FLOAT;
DEF MT GenTime GENERALIZED TIME; DEF MT Int32 INT 32;
DEF MT OctStr OCTET STRING 32; DEF MT OctStrVar VAR OCT 32;
DEF MT VisStr512 VAR VISIBLE STRING 512; DEF MT VisStr32 VISIBLE STRING 32;
DEF MT VisStrVar32 VAR VIS STR 32;
! define complex mms named types on example_vmd DEF MT IntArray NO
 DELETABLE, ARRAY [6] of Int32;
DEF MT ArrayOfIntArray ARRAY [6] of IntArray;
DEF MT Nested_Struct STRUCT X IntArray; Y Int32; END;
! define simple application named types on example_vmd DEF AT BCD FROM MT
 BCD UNSIGNED 32;
DEF AT BinTimeDateVMS FROM MT BinTimeDate, VMS ABSOLUTE TIME;
DEF AT BinTimeNoDateVMS FROM MT BinTimeNoDate, VMS ABSOLUTE TIME; DEF AT
 BitStr FROM MT BitStr, BIT STRING 32;
DEF AT BitStrBoolAry FROM MT BitStr, BOOLEAN ARRAY 32;
DEF AT BitStrVarWC FROM MT BitStrVar, WORD COUNTED STRING 32;
DEF AT Boolean FROM MT Boolean, BOOLEAN; DEF AT F_Float FROM MT Float
 F_FLOAT;
DEF AT GenTimeVMS FROM MT GenTime, VMS ABSOLUTE TIME; DEF AT Int32 FROM MT
 Int32, INTEGER 32;
DEF AT OctStr FROM MT OctStr, STRING 32;
```

```
DEF AT OctStrVarWC FROM MT OctStrVar, WORD COUNTED STRING 32; DEF AT
 VisStrWC FROM MT VisStr512, WORD COUNTED STRING 512; DEF AT VisStrVar32
 FROM MT VisStr32, STRING 32;
DEF AT VisStrVarNT FROM MT VisStrVar32, NULL TERM STRING 32; DEF AT
 VisStrVarWC FROM MT VisStrVar32, WORD COUNTED STRING 32;
! define complex application named types on example_vmd DEF AT IntArray
 FROM MT IntArray, ARRAY [0..5] of Int32;
DEF AT ArrayOfIntArray FROM MT ArrayOfIntArray ARRAY [0..5] of IntArray;
 DEF AT Nested_Struct FROM MT Nested_Struct,
STRUCT (X,X) IntArray; (Y,Y) Int32; END;
! define alternate access data types on example_vmd
DEF AT IntArray_Alt FROM MT IntArray, ARRAY [3..5] of Int32; DEF AT
 Nested_Struct_Alt1 FROM MT Nested_Struct,
STRUCT (Y,Y) Int32; END;
DEF AT Nested_Struct_Alt2 FROM MT Nested_Struct,
STRUCT (X,X) IntArray_Alt; (Y,Y) Int32; END;
! define two unnamed variables on example_vmd
DEFINE UNNAMED VARIABLE example_vmd:Longword_Unnamed
APPLICATION TYPE %:OMNI$ULONG, NUMERIC ADDRESS %X00004000, NO SUPPLY TYPE
 SPEC;
DEF UV Int32 AT Int32, NUM ADDR %X00004000;
! define three named variables on domain Dom1 of example_vmd
DEFINE NAMED VARIABLE example_vmd:Dom1(Named Variable:BinTimeDateVMS)
 DESCRIPTION "VMS date & time mapped to binary time", APPLICATION TYPE
 example_vmd:BinTimeDateVMS, NO DELETABLE;
DEF NV :Dom1(NV:Bit_String_32) AT %:OMNI$BIT32;
! set the scope to example_vmd:Dom1 set scope example_vmd:Dom1
DEF NV Nested_Struct AT example_vmd:Nested_Struct;
! display example_vmd and all dependent objects SHOW DEF example_vmd>;
! commit all definitions to the database commit;
! save the definitions to an ODF command file WRITE DEF example_vmd> to
 example_vmd.com
! delete example_vmd and all dependent objects DEL DEF example_vmd>;
! execute the command file created to re-define example_vmd
 @example_vmd.com
```

Glossary of VSI OMNI Terms

This glossary defines VSI OMNI terms that are used in this document or in reference to VSI OMNI functions.

AE	Application Entity.
API	Application Program Interface.
ASE	Application Service Element.
AST	OpenVMS Asynchronous System Trap.
AT	Application Type or Application Named Type.
CS	Companion Standard.
DOM	Domain.
FD	File Descriptor.
IOSB	Input/Output Status Block.
MMS	Manufacturing Message Specification.
MT	MMS Type or MMS Named Type.
NC	Numeric Controller.
NV	Named Variable
ODF	Omni Definition Facility.
ODS	Omni Directory Services.
PDU	Protocol Data Unit.
PI	Program Invocation.
PID	Entry in a PI list of domains.
PLC	Programmable Logic Controller.
RMS	OpenVMS Record Management System.
UV	Unnamed Variable.
VAR	Simple Variable — Named or Unnamed.
VMD	Virtual Manufacturing Device.