COM, Registry, and Events for HP OpenVMS Developer's Guide

July 2004

This document contains information about the Component Object Module (COM) for OpenVMS, the OpenVMS Registry, and OpenVMS Events logging. It also includes information about OpenVMS and Windows authentication and interoperation.

Revision/Update Information:	This manual supersedes the *COM, Registry, and Events for OpenVMS Developer's Guide, Version 1.3*.
Software Version:	COM Version 1.4 for OpenVMS
Operating System:	OpenVMS Alpha Version 7.3-1 or higher Microsoft Windows NT 4.0 SP5 or higher, or Microsoft Windows 2000 SP4 or higher

Hewlett-Packard Company
Palo Alto, California

Printed in the U.S.

ZK6539

This document was prepared using DECdocument, Version 3.3-1b.

Contents

8 Authentication

9 Active Template Library

10 COM for OpenVMS and DLL Surrogates

11 COM for OpenVMS and IEEE Floating Point

Part II OpenVMS Registry

12 Overview of OpenVMS Registry

13 OpenVMS Registry System Management

14 OpenVMS Registry Server Management

Part III OpenVMS Events

15 OpenVMS Events

Part IV Appendixes

A MIDL Compiler Options

B Troubleshooting

C Cookbook Examples: Building a Sample Application on OpenVMS

F Lists of Differences, APIs, and Interfaces

G List of Files Installed by COM for OpenVMS

H Glossary

I Acronyms

Index

Examples

Figures

Tables

Preface

Intended Audience

This document is designed primarily for developers who want to use the OpenVMS infrastructure to develop applications that move easily between the OpenVMS and Windows environments. These developers include the following:

- COM for OpenVMS developers: those who are encapsulating existing OpenVMS applications or data, as well as those who are creating new COM applications for OpenVMS systems.
- OpenVMS Registry developers: those who want to use the OpenVMS Registry to store information about their OpenVMS systems alone, or who want to use the OpenVMS Registry as a shared repository for both OpenVMS and Windows Registry information.

This document is not intended as an introduction to COM or the Registry. It assumes that readers are already familiar with object-oriented (OO) concepts and COM development techniques, as well as how the Registry works on a Windows system. The document does provide pointers to online information about COM and the Registry, and recommends other books about COM, OO development, and the Registry.

Document Structure

This document, formerly titled *OpenVMS Connectivity Developer Guide*, contains all the information you need to develop COM for OpenVMS applications and use the OpenVMS Registry. The document is divided into the following sections:

- Release notes

  COM for OpenVMS release notes.

- Part I

  COM for OpenVMS information, which includes installing, configuring, and running COM for OpenVMS, and how to develop COM for OpenVMS applications. This part also includes information about authenticating users and applications between OpenVMS and Windows systems, and information about the Active Template Library (ATL), how to develop ATL applications on COM for OpenVMS, information about DLL host surrogates, and using COM for OpenVMS with IEEE floating-point values.

- Part II

  OpenVMS Registry information, including OpenVMS Registry overview and concepts, OpenVMS Registry server startup and system management, and OpenVMS Registry server management.

- Part III

  OpenVMS Events information.

- Part IV

  Reference information, including MIDL compiler information, COM for OpenVMS cookbook examples, COM APIs supported by COM for OpenVMS, how to upgrade from previous versions of COM for OpenVMS, how to run in unauthenticated mode, lists of installed files, a glossary, and a list of acronyms.

- Index

Related Documents

For additional information about HP OpenVMS products and services, visit the following World Wide Web address:

`http://www.hp.com/go/openvms`

Reader's Comments

HP welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@hp.com**
Postal Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

How to Order Additional Documentation

For information about how to order additional documentation, visit the following World Wide Web address:

`http://www.hp.com/go/openvms/doc/order`

Conventions

The following conventions are used in this manual:

Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.) In the HTML version of this document, this convention appears as brackets, rather than a box.

. . .	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
|	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold type	Bold type represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic type	Italic type indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TYPE	Uppercase type indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Example`	This typeface indicates code examples, command examples, and interactive screen displays. In text, this type also identifies URLs, UNIX commands and pathnames, PC-based commands and folders, and certain elements of the C programming language.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

1
COM for OpenVMS Release Notes

The release notes in this chapter apply to COM Version 1.4 for OpenVMS.

1.1 COM for OpenVMS Versions

This section describes the versions of COM for OpenVMS.

- **COM Version 1.4 for OpenVMS** (this release)

  COM Version 1.4 for OpenVMS provides an optimized implementation of the underlying Win32 API infrastructure used by the COM runtime.

  COM Version 1.4 also provides corrections to problems found in previous releases. See Section 1.3 for a complete list.

- **COM Version 1.3 for OpenVMS**

  COM Version 1.3 for OpenVMS upgraded the source base for the MIDL compiler to Microsoft's MIDL compiler version 3.1.76.

  COM Version 1.3 for OpenVMS provided major enhancements at the communication protocol level for improved Windows 2000 interoperability. (Windows 2000 SP3, with the fix from Knowledge Base Article Q325409, is required.)

  COM Version 1.3 provided a command procedure to define shortcuts for COM commands, and a method to view the currently installed version. See Section 4.8 and Section 4.9 for more information.

- **COM Version 1.2 for OpenVMS**

  COM Version 1.2 for OpenVMS upgraded the source base to Windows NT 4.0 SP5. Previous versions of COM for OpenVMS were based on SP3.

  COM Version 1.2 for OpenVMS provided full support for IEEE floating-point values for your COM for OpenVMS applications. See Chapter 11 for more information.

  COM Version 1.2 for OpenVMS included error mapping capability. See Section 7.6 for more information about translating your error codes to text.

  COM Version 1.2 for OpenVMS removed all C compiler requirements for building applications. See Section 1.2.3 and Section 7.2.3 for more information.

  COM Version 1.2 for OpenVMS also provided the ability to run components in DLL surrogate processes. See Chapter 10 for more information.

- **COM Version 1.1-B for OpenVMS**

  COM Version 1.1-B for OpenVMS provided an option that allows you to choose between running COM applications in the default authenticated mode (using NTLM security features) and running COM applications in an unauthenticated mode. See Appendix E for more information.

COM Version 1.1-B for OpenVMS also provided the Active Template Library (ATL) Version 3.0 for simpler development of COM applications on OpenVMS. See Chapter 9 for more information.

As of COM Version 1.1-B for OpenVMS, the DCOM-MIDL license is no longer required.

- **COM Version 1.1-A for OpenVMS**

  COM Version 1.1-A for OpenVMS was a maintenance release that fixed a number of problems in COM Version 1.1 for OpenVMS.

- **COM Version 1.1 for OpenVMS**

  COM Version 1.1 for OpenVMS was an authenticated implementation that utilized the NTLM security features that are part of OpenVMS Version 7.2-1. This release required OpenVMS Version 7.2-1 or higher. For a list of security differences between an unauthenticated implementation and an authenticated implementation, see Table 1–1.

- **COM Version 1.0 for OpenVMS**

  COM Version 1.0 for OpenVMS was an unauthenticated implementation that did not utilize NTLM security. This release required OpenVMS Version 7.2 or higher.

Table 1–1 Summary of Security Differences

Area	Unauthenticated COM (V1.0, V1.1-B, V1.2, V1.3, V1.4)	Authenticated COM (V1.1, V1.1-A, V1.1-B, V1.2, V1.3, V1.4)
Client requests	Authenticated on Windows; not authenticated on requests to OpenVMS.	Authenticated on Windows and OpenVMS.
Security	Servers can run with the client's identity on Windows and with a prespecified OpenVMS identity on OpenVMS.	Servers can run with the client's identity on Windows and on OpenVMS.
Security	Per-method security is allowed on Windows, but only processwide security is allowed on OpenVMS.	Per-method security is allowed on Windows and on OpenVMS.
Outbound COM requests	Authenticated on Windows only.	Authenticated on Windows and OpenVMS.
Registry access	*On Windows:* controlled by Windows credentials.	*On Windows:* controlled by Windows credentials.
	On OpenVMS: relies on OpenVMS security controls such as privileges or rights identifiers.	*On OpenVMS:* controlled either by Windows credentials or by OpenVMS security controls.
Event logging	Windows only.	Windows and OpenVMS.

1.2 Upgrading from Earlier Versions of COM for OpenVMS to Version 1.4

If you are upgrading to Version 1.4 from an earlier version of COM for OpenVMS, follow the upgrade instructions in Section 4.3. In addition:

- If you are upgrading from COM Version 1.0, follow the upgrade instructions in Appendix D.

- If you are upgrading from COM Version 1.0 or COM Version 1.1, perform the tasks described in Section 1.2.1 and Section 1.2.4.
- If you are upgrading from COM Version 1.1-A, 1.1-B, 1.2, or 1.3, you do not need to perform any additional tasks.

1.2.1 Upgrading from COM Version 1.0 or 1.1 for OpenVMS Requires You to Repopulate the OpenVMS Registry

Note

If you are upgrading from any version of COM later than (but not including) COM Version 1.1 for OpenVMS, you do not need to repopulate the OpenVMS Registry.

If you are upgrading from Version 1.0 or Version 1.1 to Version 1.4, you must repopulate the OpenVMS Registry to include security settings. Use the DCOM$SETUP command to display the OpenVMS COM Tools menu, and choose option 3.

When you populate the OpenVMS Registry for COM for OpenVMS, the system prompts you to confirm the repopulation. You must answer `YES` each time. For example:

```
        [ Starting to Populate the COM for OpenVMS Registry ]

         Populating the Registry for OpenVMS may take up to 15 minutes
         depending on your system.

Enter YES to continue [default is NO]: YES

         The COM for OpenVMS Registry has already been loaded. This
         action will overwrite the current COM for OpenVMS values
         and data.

Enter YES to continue [default is NO]: YES
```

Note

Repopulating the OpenVMS Registry does not affect the registration of existing COM applications.

1.2.2 DECwindows Motif Requirement Removed

In previous versions of COM for OpenVMS, DECwindows Motif was required software. In COM Version 1.4 for OpenVMS and higher, this prerequisite is removed.

1.2.3 C Compiler Requirement Removed

In previous versions of COM for OpenVMS, DEC C Version 5.6 or higher was required software for COM for OpenVMS application development. In COM Version 1.2 for OpenVMS and higher, this prerequisite is removed. (You can still build your applications using DEC C Version 6.0 or higher; however, the C compiler is not required software.)

You can now build all applications using the recommended HP C++ compiler Version 6.5 or higher, or the minimum requirement of Compaq C++ compiler Version 6.0. (You need Compaq C++ Version 6.2-016 or higher to build ATL applications.)

For more information about building applications using the MIDL compiler, see Section 7.2.

1.2.4 Previously Registered Applications That Use Logical Names for the Local Server Path

If you previously registered any COM application using a logical name for the local server path, you must modify (reregister) the application using the actual name for the local server path.

For example, if you use the REGISTER_SIMPLE.COM command procedure to register the "Simple" application under COM Version 1.0 for OpenVMS, you must reregister the "Simple" application using the new REGISTER_SIMPLE.COM command procedure.

As of COM Version 1.1-A for OpenVMS, HP has updated the registration command files.

The system stores the COM application local server path in the OpenVMS Registry as a value data as follows:

```
"HKEY_CLASSES_ROOT\CLSID\{GUID}\LOCALSERVER32"
```

Use the following REG$CP command to modify the local server path:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE VALUE HKEY_CLASSES_ROOT\CLSID\{GUID}\Localserver32 -
_$ /TYPE=SZ/DATA=device:[directory]image-name.EXE
```

A GUID is the COM application CLSID. For more information about Localserver32 and CLSID, see Section 7.5.

1.2.5 Changes to the Examples

In COM Version 1.1-A for OpenVMS, the names of the server images in the Dispatch_Sample1 example changed. If you previously built and registered this application and you want to build the new version, you must reregister the server after it has been built.

1.3 Problems Fixed in This Release

The following notes describe problems that have been fixed in COM Version 1.4 for OpenVMS.

1.3.1 New NTA$LOGON.EXE Fixes Data Corruption in Password File

HP testing has uncovered a problem that occurs while using the DCOM$SETUP menu option number 8 to update the DCOM$RPCSS user password in the COM for OpenVMS Service Control Manager password file. This problem can potentially cause the password file to be corrupted.

A new NTA$LOGON.EXE image corrects this problem. This image can be obtained by picking up the latest DCE update kit.

1.3.2 Access Violation When Compiling Very Large IDL Files

In certain instances, an access violation would occur when compiling very large IDL files. COM Version 1.4 provides a modification to the MIDL compiler that prevents this problem.

1.3.3 Windows 2000 Interoperability Requires Windows 2000 SP4 and Latest DCERPC

As part of interoperability testing, HP has identified problems in the authentication layer that prevents authenticated COM requests between OpenVMS and Windows 2000 systems.

The solution to these problems is to install Windows 2000 SP4 on your Windows system, and the latest DCE$LIB_SHR.EXE (available from your support center) on your OpenVMS system.

1.3.4 ERROR_ACCESS_DENIED (C0000005 or 80070005)

COM for OpenVMS in authenticated mode reported intermittent ERROR_ ACCESS_DENIED (C0000005 or 80070005) errors when running an OpenVMS client to a Windows 2000 server. These errors were seen only if the domain name was fewer than 12 characters. This problem has been fixed in this release.

Under heavy load testing, HP still encounters a few remaining ERROR_ACCESS_ DENIED errors, but the occurence is less than 1%. To avoid this problem completely, use a domain name of 12-15 characters.

1.4 Known Problems in the Current Release

The following notes describe the known problems with COM Version 1.4 for OpenVMS. These problems currently do not have fixes.

1.4.1 Fatal Exception in DCOM$RPCSS While Launching Multiple Instances of COM Applications

During extensive testing by HP in unauthenticated mode, in certain instances the DCOM$RPCSS process may experience a fatal exception while trying to launch multiple instances of different COM applications. This results in all client processes reporting a 8007071C error.

To return to a working environment, shut down and restart DCOM$RPCSS. A possible workaround is to run your application as a persistent server.

HP is continuing to investigate this problem. If you encounter this problem, please contact your HP support center.

1.4.2 COM Version 1.4 Fails with Microsoft MS04-012 (KB 828741) Patch

HP testing has discovered that after installing the Microsoft security patch MS04-012 (Knowledge Base article 828741), an OpenVMS client connection to a Windows server fails. The application will fail with an access denied error such as C0000005 or 80070005.

If you have already installed this security patch and are experiencing problems, a possible workaround is to temporarily uninstall the patch.

A solution to this problem is being jointly investigated by Microsoft and HP, and an update will be provided as soon as it becomes available.

1.4.3 Kernel Threads and Upcalls Not Supported

COM for OpenVMS applications cannot be built with kernel threads or upcalls enabled. This support will be available in a future release.

1.4.4 Errors Seen Between OpenVMS Systems Running COM Version 1.3 Under Heavy Load

Testing by HP has uncovered the errors E-OUTOFMEMORY (8007000E), DCERPC-E-WHOAREYOUFAILED (EE1282FA) and CO_E_SERVER_EXEC_ FAILURE (80080005) between OpenVMS systems running heavy load tests with COM Version 1.3 for OpenVMS. This problem was not seen when the tests were run locally or between Windows and OpenVMS systems.

1.5 Limitations and Restrictions

The following sections contain general release note information.

1.5.1 Windows XP Not Supported

HP has not tested COM Version 1.4 with Windows XP. Therefore, Windows XP is not supported in COM Version 1.4 for OpenVMS.

1.5.2 NetBEUI as Listed Protocol in DCOMCNFG

If you have NetBEUI listed as a default protocol in DCOMCNFG, you might see various errors when running sample programs between OpenVMS and Windows 2000. To correct these problems, remove NetBEUI from the list of default protocols and reboot your Windows system.

1.5.3 COM Version 1.0 for OpenVMS and COM Version 1.4 for OpenVMS Not Supported in the Same Cluster

When you install and configure any version of COM for OpenVMS higher than Version 1.0 on any node in a cluster, you make clusterwide modifications to the OpenVMS Registry that prevent COM Version 1.0 for OpenVMS from running on any other node in the same cluster.

1.5.4 Threading Model Supported by COM for OpenVMS

COM Version 1.4 for OpenVMS supports only the multithreaded apartment (**MTA**, also known as **free threads**) model for application servers. The MTA model allows a component to have more than one thread. However, you must ensure that your code is thread safe.

The threading model initialization call is as follows:

```
CoInitializeEx(
   NULL,
   COINIT_MULTITHREADED
   )
```

Because `CoInitialize()` implies the single-threaded apartment (STA) model, you cannot use it in place of `CoInitializeEx()` in a server application.

1.5.5 Enhanced NTLM in Windows NT SP4 and Later Versions Not Supported

In Windows NT SP4, Microsoft introduced enhanced NTLM support. COM Version 1.4 for OpenVMS does not support enhanced NTLM.

If you want to use COM Version 1.4 for OpenVMS with SP4 or later, you must be sure that enhanced NTLM is disabled.

Ongoing testing by HP has show that, with SP4 or higher and enhanced NTLM disabled, authentication requests fail if you use passwords that are longer than 12 characters.

1.5.6 Specifying Activation Security in CoCreateInstanceEx

The `pServerInfo` parameter of the `CoCreateInstanceEx` API allows you to specify a user name and password that will be used for authentication on the remote server system. The user name and password are part of the `COAUTHIDENTITY` structure inside the `COAUTHINFO` structure, which is inside the `COSERVERINFO` structure that is passed as the `pServerInfo` parameter to `CoCreateInstanceEx`.

The current NTLM security implementation on OpenVMS does not support this feature for COM client applications on OpenVMS. This feature is supported for COM clients on Windows communicating with COM servers on OpenVMS.

1.5.7 RPC Communication Failures Caused by Advanced Server

In a cross-domain environment, under some load situations, COM applications may report errors that are a side effect of the HP Advanced Server for OpenVMS having lost a connection between domain controllers. The Advanced Server reports this error as follows:

```
NET5719:    No domain controller for the domain 'xxxxx' is available.
```

A series of these events over a limited time interval may lead to COM applications reporting RPC communications failures (%x8007071c). In this situation, a stop and start of DCOM$RPCSS may be required to clear the error.

See Section 5.4.6 for more information. If the NET5719 events persist, contact your HP support center.

1.5.8 RPC Cannot Support Failure (800706E4) Error Message

If you attempt to use the single-threaded apartment (STA) model, some COM APIs may display the following return status code:

```
(800706E4)
```

COM Version 1.4 for OpenVMS does not support the STA model. For more information, see Section 1.5.4.

2
OpenVMS Registry Release Notes

For the latest OpenVMS Registry information, refer to the *OpenVMS Release Notes* for the current version of the operating system.

Part I

COM for OpenVMS

The following chapters provide an overview of COM for OpenVMS, provide instructions for installing and configuring COM for OpenVMS and related software, and describe and explain how to create COM applications using COM for OpenVMS.

3
Overview of COM for OpenVMS

3.1 What is COM?

Component Object Model (COM) is a technology from Microsoft that lets developers create distributed network objects. First introduced by Microsoft in its Windows 3.*x* product, COM was initially called Object Linking and Embedding (OLE). COM provides a widely available, powerful mechanism for customers to adopt and adapt to a new style multivendor distributed computing, while minimizing new software investment.

The former Digital Equipment Corporation (now part of Hewlett-Packard Company) and Microsoft jointly developed the COM specification. First released as NetOLE (Network OLE) and then renamed DCOM (Distributed COM), the COM specification now includes network functionality. That is, COM now supports distributed network objects.

COM is an object-based programming model designed to promote software interoperability. COM allows two or more applications (or components) to cooperate with one another easily, even if the objects are written by different vendors at different times and in different programming languages, or if they are running on different machines with different operating systems. To support its interoperability features, COM defines and implements mechanisms that allow applications to connect to each other as software objects.

COM implementations are available on Windows 2000, Windows NT, Windows 98, Windows 95, OpenVMS, and HP *Tru64*™ UNIX, as well as other UNIX platforms.

3.1.1 Suggested Reading

The following resources can provide you with more information on COM and related topics:

- Third-party books on COM:
  - *COM Beyond Microsoft/Designing and Implementing COM Servers on Compaq Platforms*, Terence P. Sherlock and Gene Cronin, Digital Press, Boston, MA, 2000. ISBN: 1-55558-226-5.
  - *Inside COM/Microsoft's Component Object Model*, Dale Rogerson, Microsoft Press, Redmond, WA, 1997. ISBN: 1-57231-349-8.

    The examples in this document are taken from Dale Rogerson's book and are used with the publisher's permission.
  - *Essential COM*, Don Box, Addison Wesley Longman, Reading, MA, 1998. ISBN: 0-201-63446-5.
  - *Effective COM*, Don Box, Keith Brown, Tim Ewald, and Chris Sells, Addison Wesley Longman, Reading, MA, 1998. ISBN: 0-201-37968-6.

— *DCOM Explained*, Rosemary Rock-Evans, Digital Press, Woburn, MA, 1998. ISBN: 1-55558-216-8.

Provides a good introduction to DCOM and COM, and discusses COM implementations on various platforms.

— *Understanding ActiveX and OLE*, David Chappell, Microsoft Press, Redmond, WA, 1996. ISBN: 1-57231-216-5.

- Websites:

— *The Component Object Model Specification*, available from the Microsoft COM website:

```
http://www.microsoft.com/com
```

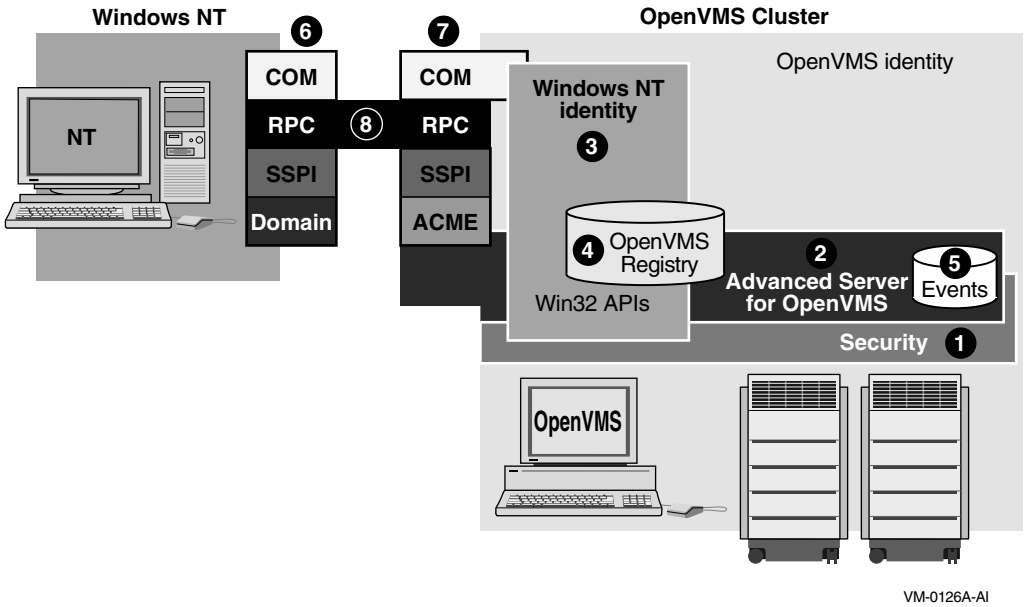
3.2 Overview of COM for OpenVMS

COM for OpenVMS is HP's implementation of Microsoft's Windows NT 4.0 Service Pack 5 (SP5) Component Object Model (COM) software on the OpenVMS Alpha operating system.

In support of COM for OpenVMS, HP ported Windows NT infrastructure to OpenVMS, including a registry, event logger, NTLM security, and Win32 APIs. COM for OpenVMS is layered on The Open Group's Distributed Computing Environment (DCE) RPC. COM for OpenVMS supports communication among objects on different computers on a local area network (LAN), a wide area network (WAN), or the Internet. COM for OpenVMS is important to the Affinity for OpenVMS program because it delivers a key piece of connectivity with Windows.

Figure 3–1 shows the OpenVMS infrastructure.

Figure 3–1 OpenVMS Infrastructure and COM for OpenVMS



In Figure 3–1 the key pieces of the OpenVMS infrastructure are as follows:

Windows system

The smaller box on the left side of Figure 3–1 represents the Windows system.

OpenVMS Cluster/OpenVMS identity

The large box on the right side of Figure 3–1 represents the OpenVMS system. Within and around this box you can see several other boxes labeled with numbers. The following list describes these numbered items:

❶ OpenVMS security

This is the standard OpenVMS security (login, authentication, ACLs, and so on) available with all OpenVMS systems.

❷ HP Advanced Server for OpenVMS

The HP Advanced Server for OpenVMS provides authentication of Windows users to OpenVMS and provides a connection to the OpenVMS Registry and events viewer for Windows users.

❸ Windows identity/Win32 APIs

The OpenVMS Security, MSV1_0 ACME agent, HP Advanced Server for OpenVMS, OpenVMS Registry, event logger, and Win32 APIs (COM APIs) all contribute to the creation of a Windows identity within the OpenVMS system.

❹ OpenVMS Registry

The OpenVMS Registry, like the registry on Windows systems, allows you to store system, software, and hardware configuration information on OpenVMS. COM for OpenVMS uses the OpenVMS Registry to store information about COM applications. For detailed information about the OpenVMS Registry, see Part II of this document.

❺ Event logger

Like the event logger on Windows systems, the event logger on OpenVMS records informational, warning, and error messages about COM events. For detailed information about the OpenVMS Events, see Chapter 15.

❻ Windows COM stack

On the Windows system, COM requests and responses pass through the COM, RPC, SSPI (security), and Domain layers.

❼ OpenVMS COM stack

The OpenVMS system mirrors the Windows COM stack, with some additions. On the OpenVMS system, COM requests and responses pass through the COM, RPC, SSPI (security), MSV1_0 ACME agent, and HP Advanced Server for OpenVMS layers. The MSV1_0 ACME agent (shown as ACME in Figure 3–1) is an extension to the Authentication and Credential and Management (ACM) authority. Authentication is explained in detail in Chapter 8.

❽ Connection through RPC layer

The COM connection between the Windows system and OpenVMS is always through the RPC layer.

For developers, the COM for OpenVMS developer's kit provides a Microsoft Interface Definition Language (MIDL) compiler and C-style header files for application development. For more information about the OpenVMS MIDL compiler, see Section F.1.4.1.

OpenVMS includes a utility to acquire Windows credentials if you are running COM for OpenVMS in authenticated mode. For more information about getting Windows credentials through NTA$LOGON, see Section 5.1 and Chapter 8.

COM for OpenVMS also provides a free run-time environment on OpenVMS Alpha for the deployment of COM for OpenVMS client and server applications.

You can find a complete description of Microsoft's COM, including protocol specifications and programming documentation, at the Microsoft COM website at the following location:

```
www.microsoft.com/com
```

The COM for OpenVMS implementation is a subset of the full Microsoft COM implementation. For a complete list of the COM for OpenVMS APIs, supported interfaces, and implementation differences, see Appendix F.

While general interest in COM continues to grow, COM remains a sophisticated technology. It is not aimed at the naive user, but rather at skilled programmers, such as independent software vendors (ISVs) and large management information system (MIS) shops.

3.2.1 How COM for OpenVMS Uses the OpenVMS Registry

COM for OpenVMS requires the OpenVMS Registry. Like its registry database counterpart on Windows systems, the OpenVMS Registry stores information about COM applications—specifically those COM applications running on OpenVMS. These COM for OpenVMS applications use the OpenVMS Registry to store CLSIDs (class IDs), startup information, security settings, and so on in the OpenVMS Registry database. COM for OpenVMS uses the Win32 APIs implemented on OpenVMS to read and write this information to the OpenVMS Registry.

COM for OpenVMS requires access to the OpenVMS Registry database. If COM for OpenVMS cannot access the OpenVMS Registry, COM for OpenVMS will not start. For more information about the OpenVMS Registry, see Chapter 12.

3.3 Using COM for OpenVMS

You can use COM for OpenVMS to do the following:

- Develop new COM for OpenVMS COM applications
- Encapsulate existing applications for use with COM for OpenVMS

The following sections discuss new application development and encapsulation in more detail.

An example of a COM application to encapsulate an existing OpenVMS application is included with this release. The example can be found in DCOM$EXAMPLES:[WRAPPER]. A README file describes the example and how to build it.

3.3.1 Developing New Applications

Your organization might use COM for OpenVMS to develop new applications under the following circumstances:

- You want to share data between an OpenVMS server and Windows clients in a two-tier client/server computing model.

- You want to share data and to place business logic in the middle tier of a three-tier computing environment.

  For example, you might have a Windows system as the client so you can take advantage of its graphical user interface. You could write business logic as a collection of COM objects on a middle-tier server; while the third-tier large-capacity, high-availability OpenVMS server provides database access.

- You want to share data between one or more OpenVMS systems or between OpenVMS and other non-Windows systems using COM.

The advantages of using COM for OpenVMS include:

- COM for OpenVMS provides a good programming model for programmers with C++ and object-oriented programming skills.
- COM for OpenVMS provides multivendor interoperability. COM is a standard with implementations available on a number of platforms today, and ports for additional platforms are in development.
- The COM for OpenVMS run-time provides automated data marshaling and unmarshaling.
- COM provides OLE Automation services to support communications with Microsoft Visual Basic® applications. Visual Basic is a very popular programming environment for client/server computing.
- COM provides version support for components so you can upgrade applications over time without breaking existing environments.

See Chapter 7 and Appendix C for examples of developing COM for OpenVMS applications.

3.3.2 Encapsulating Existing Applications

If you have monolithic applications written in procedural languages (such as Fortran and COBOL) with character-cell interfaces, you can put a COM "wrapper" or jacket around these applications to allow them either to run on new platforms or to remain on OpenVMS and run in a client/server environment.

The risk associated with completely reengineering some older applications is high. Many applications are large, complex, poorly documented, and not well understood by their current maintainers. Encapsulating a legacy application can be less risky than reengineering and can be the first step in a rewrite. Over time, pieces of the legacy application can be rewritten, while the older version of the application remains stable and available. Encapsulation also allows developers to reuse code, saving time and resources.

Disadvantages to encapsulation include more complex maintenance efforts and the inability to make changes to the underlying code. If the legacy application was unstable or hard to maintain, the encapsulated application will not be any better, and might be made worse because of the wrapper.

There are several layers of a traditional procedural application that you can encapsulate: the user interface (UI), the database, and the data manipulation routines.

— User interface

  If you choose to encapsulate the user interface, the UI could be supported on some other platform (for example, from a graphical user interface [GUI] on a Windows system).

Encapsulating and moving the UI to the user's desktop can mean that the rest of the application remains on OpenVMS. Batch processing programs are well suited to user interface encapsulation. Applications that do screen management (for example, SMG or FMS) could have their older character-cell interface encapsulated using COM for OpenVMS, providing users access through newer Windows style dialog boxes.

— Database

If you choose to encapsulate a database using COM for OpenVMS, the database could be accessed from parts of a distributed application running on other platforms. The advantage of this approach is that the programmer can keep the database on OpenVMS (a stable, 24x365 system), while the user interface and data access routines are on remote (and perhaps less reliable) systems.

— Database manipulation routines

If you choose to encapsulate the database manipulation routines, the routines could be accessed from any other COM component in a heterogeneous computing environment.

Encapsulating an OpenVMS application using COM for OpenVMS means that you write a COM for OpenVMS server that talks to the application being encapsulated. The COM for OpenVMS server passes arguments to the application in the order and format that the application expects. The COM for OpenVMS server then intercepts the output from the application and directs it to the display device, user interface, or other routines.

4
Installing the COM for OpenVMS Kit

This chapter provides a list of the contents of the COM for OpenVMS kit, a list of prerequisite software, and preinstallation requirements. It also describes how to install COM for OpenVMS and includes postinstallation instructions.

4.1 Contents of the COM Version 1.4 for OpenVMS Kit

COM Version 1.4 for OpenVMS contains the following:

- Software
  - — COM for OpenVMS run-time libraries
  - — COM for OpenVMS MIDL compiler and header files
  - — COM for OpenVMS configuration utilities
  - — Active Template Library Version 3.0
  - — Sample applications
- Documentation
  - — *COM, Registry, and Events for HP OpenVMS Developer's Guide* (in PostScript, HTML, and PDF formats)

4.2 Prerequisites

The following software is required:

- For OpenVMS systems
  - — OpenVMS Version 7.3-1 or higher
  - — For Windows 2000 Interoperability:
    The most recent DCE$LIB_SHR.EXE (available from your support center).
  - — For COM for OpenVMS application development:
    Recommended: HP C++ Version 6.5 or higher
    Minimum requirement: Compaq C++ Version 6.0 or higher

    To build ATL applications on OpenVMS:
    Compaq C++ Version 6.2-016 or higher
  - — HP TCP/IP Services for OpenVMS Version 5.0 or equivalent
  - — HP Advanced Server for OpenVMS Version 7.3 or higher

    (HP Advanced Server for OpenVMS is not required if you are running COM for OpenVMS in unauthenticated mode.)

— Before installing COM for OpenVMS check that you have the required free global pages, global sections, and disk blocks. The following table lists the requirements.

Software	Global pages	Global sections	Disk blocks
COM for OpenVMS	11,000	27	34000
RPC Runtime	3,300	14	N/A

For Advanced Server requirements: See the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.
For TCP/IP requirements: See the *HP TCP/IP Services for OpenVMS Installation and Configuration* document.

- For Windows systems
  - — Windows NT 4.0 with Service Pack 5 or higher installed

    OR
  - — Windows 2000 with Service Pack 4 or higher installed
  - — Microsoft Visual C++ or Visual Basic (for Windows client development and information about MIDL compiler). See the Microsoft website for compiler version requirements.
  - — TCP/IP enabled (needed for OpenVMS connectivity)

4.3 Supported COM for OpenVMS Installations

The following sections describe COM Version 1.4 for OpenVMS installation and upgrade options.

Note

If you want to run COM Version 1.4 for OpenVMS in unauthenticated mode, see Section E.1.

If you want to do this	Read this section
Install COM for OpenVMS on an OpenVMS standalone system for the first time.	Section 4.4
Install COM for OpenVMS on an OpenVMS Cluster system for the first time.	Section 4.6
Upgrade from earlier versions of COM for OpenVMS on an OpenVMS standalone system.	Section 4.5
Upgrade from earlier versions of COM for OpenVMS on an OpenVMS Cluster system.	Section 4.7

4.4 Installing COM for OpenVMS on an OpenVMS Standalone System

Use the following procedure:

1. Install OpenVMS Version 7.3-1 or higher. For this procedure, see the *OpenVMS Alpha Version 7.x Upgrade and Installation Manual.*

2. Install TCP/IP Services. For this procedure, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
3. Boot the installed system from the system disk.
4. Install COM Version 1.4 for OpenVMS. For this procedure, see Section 4.11.
5. Install HP Advanced Server for OpenVMS. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide.*
6. Configure TCP/IP Services (set up for startup and reboot); then start TCP/IP. You must configure the PWIP driver for HP Advanced Server for OpenVMS to use TCP/IP Services. For information about configuring TCP/IP, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
7. Configure the OpenVMS Registry as follows:
   - Run `REG$CONFIG.COM` to configure the OpenVMS Registry.
   - Edit the `SYLOGICALS.COM` file to define the `SYS$REGISTRY` logical as follows:

     ```
     $ DEFINE/SYSTEM SYS$REGISTRY directory-specification
     ```
8. Start OpenVMS Registry by running the `REG$STARTUP.COM` file.
9. If you want to run DCE, start DCE now.

   Note

   You do not need DCE to run COM for OpenVMS, but if your environment uses DCE, HP recommends that you start DCE now.

   For this procedure, see the *Compaq DCE Installation and Configuration Guide.*

   For more information about OpenVMS external authentication, see Section 5.1.
10. Configure HP Advanced Server for OpenVMS. You may need to reboot, depending on your system configuration. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide.*
11. Start HP Advanced Server for OpenVMS (set up for startup on reboot). For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide.*
12. Start the ACME server. Use the following command:

    ```
    $ @SYS$STARTUP:NTA$STARTUP_NT_ACME
    ```
13. Start RPC. Use the following command:

    ```
    $ @SYS$STARTUP:DCE$RPC_STARTUP.COM
    ```
14. Configure COM for OpenVMS. For this procedure, see Section 4.12 and Section 6.2.
    - Populate the OpenVMS Registry. For this procedure, see Section 6.2. Use option 3 to populate the OpenVMS Registry database.

- Create any OpenVMS and HP Advanced Server for OpenVMS accounts needed by the COM for OpenVMS Service Control Manager. For more information, see Section 6.2. Use option 8 to create the accounts.

15. Edit the SYS$MANAGER:SYLOGICALS.COM file and add the following line:

```
$ DEFINE DCOM$TO_BE_STARTED TRUE
```

16. Start COM for OpenVMS. For this procedure, see Section 4.13.

4.5 Upgrading COM for OpenVMS on an OpenVMS Standalone System

Note

Before you start, HP recommends that you disable any HP Advanced Server for OpenVMS, OpenVMS Registry, and layered product automatic startups so that these products do not start until you have upgraded COM for OpenVMS and its associated components.

Use the following procedure:

1. Edit the SYLOGICALS.COM file to stop the following products from starting:
   - OpenVMS Registry (comment the line DEFINE/SYSTEM REG$TO_BE_STARTED TRUE)
   - COM for OpenVMS (comment the line DEFINE DCOM$TO_BE_STARTED TRUE)
2. Edit the SYS$STARTUP:SYSTARTUP_VMS.COM file to stop the following products from starting:
   - HP Advanced Server for OpenVMS (comment the line @SYS$STARTUP:PWRK$STARTUP.COM).

If COM for OpenVMS is currently running, shut down COM for OpenVMS first, HP Advanced Server for OpenVMS (if running), and then the OpenVMS Registry.

Use the following procedure:

1. Upgrade to OpenVMS Version 7.3-1 or higher. For this procedure, see the *OpenVMS Alpha Version 7.x Upgrade and Installation Manual.*
2. If you need to upgrade TCP/IP, upgrade TCP/IP now. For this procedure, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
3. Boot the upgraded system from the system disk.
4. Upgrade COM for OpenVMS. For this procedure, see Section 4.11.
5. Install or upgrade HP Advanced Server for OpenVMS. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide.*
6. Start TCP/IP unless you have enabled TCP/IP to start on a reboot. For this procedure, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.

7. Start the OpenVMS Registry unless you have enabled the OpenVMS Registry to start on a reboot.
8. If you want to run DCE, start DCE now.

   Note

   You do not need DCE to run COM for OpenVMS, but if your environment uses DCE, HP recommends that you start DCE now.

   For this procedure, see the *Compaq DCE Installation and Configuration Guide*.

   For more information about OpenVMS external authentication, see Section 5.1.
9. Configure HP Advanced Server for OpenVMS. You may need to reboot, depending on your system configuration. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.
10. Start HP Advanced Server for OpenVMS (set up for startup on reboot). For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.
11. Start the ACME server. Use the following command:

    ```
    $ @SYS$STARTUP:NTA$STARTUP_NT_ACME
    ```
12. Start RPC. Use the following command:

    ```
    $ @SYS$STARTUP:DCE$RPC_STARTUP.COM
    ```
13. See Appendix D for detailed information about upgrading from COM Version 1.0 for OpenVMS to COM Version 1.4 for OpenVMS.
14. Configure COM for OpenVMS. For this procedure, see Section 4.12 and Section 6.2.
    - Populate the OpenVMS Registry. For this procedure, see Section 6.2. Use option 3 to populate the OpenVMS Registry database.
    - Create any OpenVMS and HP Advanced Server for OpenVMS accounts needed by the COM for OpenVMS Service Control Manager. For more information, see Section 6.2. Use option 8 to create the accounts.
15. Edit the `SYLOGICALS.COM` file and add the following line:

    ```
    $ DEFINE DCOM$TO_BE_STARTED TRUE
    ```
16. Start COM for OpenVMS. For this procedure, see Section 4.13.

4.6 Installing COM for OpenVMS on an OpenVMS Cluster

Note

This cluster installation procedure assumes you are installing COM for OpenVMS on a single system disk.

Use the following procedure:

1. Install OpenVMS Version 7.3-1 or higher on all system disks as required. For this procedure, see the *OpenVMS Alpha Version 7.x Upgrade and Installation Manual*.
2. Install TCP/IP. For this procedure, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
3. Boot the installed system from the system disk.
4. Install COM Version 1.4 for OpenVMS. For this procedure, see Section 4.11.
5. Install HP Advanced Server for OpenVMS on this node in the cluster. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.

   Note

   You must install HP Advanced Server for OpenVMS on at least one Alpha node in the cluster. On the other nodes, you can either install HP Advanced Server for OpenVMS or select `External Authentication images (only)`.

6. Configure TCP/IP (set up for startup on reboot on each node) and start TCP/IP. You must configure the PWIP driver for HP Advanced Server for OpenVMS to use TCP/IP. For information about configuring TCP/IP, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
7. Configure the OpenVMS Registry:
   - Run `REG$CONFIG.COM` to configure the OpenVMS Registry. You need to configure the OpenVMS Registry only once for the cluster.
   - Set the `SYS$REGISTRY` logical to `DEFINE/SYSTEM` on every Alpha node in the cluster that will run the OpenVMS Registry server.
   - Edit the `SYLOGICALS.COM` file on every node in the cluster as follows:
     — If the cluster uses a single, cluster-common `SYLOGICALS.COM` file that is called by each node's `SYLOGICALS.COM` file, you do not need to make any changes.
     — On those nodes where you do not want the OpenVMS Registry server to run, add the following line to the `SYLOGICALS.COM` file:

     ```
     $ DEFINE/SYSTEM REG$TO_BE_STARTED FALSE
     ```

     HP Advanced Server for OpenVMS requires that the OpenVMS Registry be running on a node in the cluster.
8. Configure DCE.

   Note

   You do not need DCE to run COM for OpenVMS, but if your environment uses DCE, HP recommends that you start DCE now.

For this procedure, see the *Compaq DCE Installation and Configuration Guide*.

9. If you want to run DCE, start DCE now. You must configure DCE on each node on which you want to run DCE.

   For more information about OpenVMS external authentication, see Section 5.1.

10. Configure and start HP Advanced Server for OpenVMS. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.

    If this node is running HP Advanced Server for OpenVMS, set up HP Advanced Server for OpenVMS for startup on reboot (edit the `SYS$MANAGER:SYSTARTUP_VMS.COM` file as necessary).

    If this node is not running HP Advanced Server for OpenVMS, edit the `SYLOGICALS.COM` file and define the `PWRK$ACME_SERVER` logical. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*. For more information about the `PWRK$ACME_SERVER` logical, see Table 8–2.

11. Start the ACME server. Use the following command:

    ```
    $ @SYS$STARTUP:NTA$STARTUP_NT_ACME
    ```

12. Start RPC. Use the following command:

    ```
    $ @SYS$STARTUP:DCE$RPC_STARTUP.COM
    ```

13. Configure COM for OpenVMS. For this procedure, see Section 4.12 and Section 6.2.

    - Populate the OpenVMS Registry. For this procedure, see Section 6.2. Use option 3 to populate the OpenVMS Registry database. You need to populate the OpenVMS Registry only once in a cluster.
    - Create any OpenVMS and HP Advanced Server for OpenVMS accounts needed by the COM for OpenVMS Service Control Manager. For more information, see Section 6.2. Use option 8 to create the accounts. You need to create these accounts only once in a cluster.

14. Edit the `SYLOGICALS.COM` file and add the following line:

    ```
    $ DEFINE DCOM$TO_BE_STARTED TRUE
    ```

15. Start COM for OpenVMS. For this procedure, see Section 4.13.

4.7 Upgrading COM for OpenVMS in an OpenVMS Cluster

Note

This cluster upgrade procedure assumes you are installing COM for OpenVMS on a single system disk.

Note

Before you start, HP recommends that you disable any HP Advanced Server for OpenVMS and layered products automatic startups so these products do not start until you have upgraded COM for OpenVMS and its associated components.

Use the following procedure:

1. Edit the SYLOGICALS.COM file to stop the following products from starting:
   - OpenVMS Registry (comment the line DEFINE/SYSTEM REG$TO_BE_STARTED TRUE)
   - COM for OpenVMS (comment the line DEFINE DCOM$TO_BE_STARTED TRUE)
2. Edit the SYS$STARTUP:SYSTARTUP_VMS.COM file to stop the following products from starting:
   — HP Advanced Server for OpenVMS (comment the line @SYS$STARTUP:PWRK$STARTUP.COM)

If COM for OpenVMS is currently running, shut down COM for OpenVMS first, HP Advanced Server for OpenVMS (if running), and then the OpenVMS Registry on all nodes in the cluster.

Use the following procedure:

1. Upgrade to OpenVMS Version 7.3-1 or higher on all required system disks. For this procedure, see the *OpenVMS Alpha Version 7.x Upgrade and Installation Manual.*
2. Upgrade TCP/IP. For this procedure, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
3. Boot the upgraded system from the system disk.
4. Upgrade to COM Version 1.4 for OpenVMS. For this procedure, see Section 4.11.
5. Upgrade HP Advanced Server for OpenVMS on this node in the cluster. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide.*

Note

You must install HP Advanced Server for OpenVMS on at least one Alpha node in the cluster. On the other nodes, you can either install HP Advanced Server for OpenVMS or select External Authentication images (only).

---

6. Configure TCP/IP (set up for startup on reboot on each node). You must configure the PWIP driver for HP Advanced Server for OpenVMS to use TCP/IP. For information about configuring TCP/IP, see the *HP TCP/IP Services for OpenVMS Installation and Configuration* manual or your TCP/IP supplier's documentation.
7. Configure the OpenVMS Registry as follows:
   - Run REG$CONFIG.COM to configure the OpenVMS Registry. You need to configure the OpenVMS Registry only once for the cluster.

- Edit the SYLOGICALS.COM file on every node that will run the OpenVMS Registry server to define the SYS$REGISTRY logical. For example:

  ```
  $ DEFINE/SYSTEM SYS$REGISTRY cluster-visible-directory-specification
  ```

  Edit the SYLOGICALS.COM file on every node in the cluster as follows:

  — If the cluster uses a single, cluster-common SYLOGICALS.COM file that is called by each node's SYLOGICALS.COM file, you do not need to make any changes.

  — On those nodes where you do not want the OpenVMS Registry server to run, add the following line to the SYLOGICALS.COM file:

  ```
  $ DEFINE/SYSTEM REG$TO_BE_STARTED FALSE
  ```

8. Configure and start HP Advanced Server for OpenVMS. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*.

   If this node is running HP Advanced Server for OpenVMS, set up HP Advanced Server for OpenVMS for startup on reboot (edit the SYS$STARTUP file as necessary).

   If this node is not running HP Advanced Server for OpenVMS, edit the SYLOGICALS.COM file and define the PWRK$ACME_SERVER logical. For this procedure, see the *HP Advanced Server for OpenVMS Server Installation and Configuration Guide*. For more information about the PWRK$ACME_SERVER logical, see Table 8–2.

9. Start the ACME server. Use the following command:

   ```
   $ @SYS$STARTUP:NTA$STARTUP_NT_ACME
   ```

10. Start RPC. Use the following command:

    ```
    $ @SYS$STARTUP:DCE$RPC_STARTUP.COM
    ```

11. See Appendix D for detailed information about upgrading from COM Version 1.0 for OpenVMS to COM Version 1.4 for OpenVMS.

12. Configure COM for OpenVMS. For this procedure, see Section 4.12 and Section 6.2.

    - Populate the OpenVMS Registry. For this procedure, see Section 6.2. Use option 3 to populate the OpenVMS Registry database. You need to populate the OpenVMS Registry only once in a cluster.
    - Create any OpenVMS and HP Advanced Server for OpenVMS accounts needed by the COM for OpenVMS Service Control Manager. For more information, see Section 6.2. Use option 8 to create the accounts. You need to create these accounts only once in a cluster.

13. Edit the SYLOGICALS.COM file and add the following line:

    ```
    $ DEFINE DCOM$TO_BE_STARTED TRUE
    ```

14. Start COM for OpenVMS on a particular node. For this procedure, see Section 4.13.

4.8 Defining Shortcut Commands for COM for OpenVMS

COM for OpenVMS provides a command procedure that defines shortcut commands for many regular COM commands.

Add the following command to your login command procedure, or execute the command line after logging in:

```
$  @SYS$STARTUP:DCOM$DEFINE_COMMANDS.COM
```

SYS$STARTUP:DCOM$DEFINE_COMMANDS.COM defines the following commands:

```
$ DCOMSTA*RT    :== @SYS$STARTUP:DCOM$STARTUP.COM
$ DCOMSTO*P     :== @SYS$STARTUP:DCOM$SHUTDOWN.COM
$ DCOMSTOPN*O   :== @SYS$STARTUP:DCOM$SHUTDOWN.COM NOCONFIRM
$ DCOMSE*TUP    :== @SYS$STARTUP:DCOM$SETUP.COM
$ DCOMC*NFG     :== $SYS$SYSTEM:DCOM$CNFG.EXE
$ DCOMT*OOL     :== $SYS$SYSTEM:DCOM$TOOL.EXE
$ NTLOG*ON      :== $SYS$SYSTEM:NTA$LOGON.EXE
$ DCOMOUT       :== TYPE/PAGE SYS$MANAGER:DCOM$RPCSS.OUT
$ DCOMEV*ENTS   :== TYPE/PAGE SYS$MANAGER:DCOM$EVENTLOG.RPT
$ DCOMVER       :== $SYS$SYSTEM:DCOM$TOOL.EXE SHOW VERSION
$ DCOMER*ROR    :== $SYS$SYSTEM:DCOM$TOOL.EXE SHOW ERROR
```

4.9 Checking the COM for OpenVMS Version

With COM Version 1.3 or higher for OpenVMS, you can view the currently installed version of COM. To do so, invoke the following command:

```
$ DCOM$TOOL == "$DCOM$TOOL"
$ DCOM$TOOL SHOW VERSION
```

For more ways to use to the DCOM$TOOL Utility, see Section 7.6.2.

4.10 Understanding the COM for OpenVMS Environment

COM for OpenVMS relies on a number of interrelated servers (processes) and operating system images. In most cases, the servers start automatically when you restart the system. (Automatic startup requires that you have installed and configured each component and have made appropriate changes to the SYLOGICALS.COM file.) For more information about starting and configuring the servers, see Section 4.3.

Figure 4–1 shows the relationships and dependencies of the processes and operating system layers.

Figure 4–1 Interrelationships Among Processes and Layers

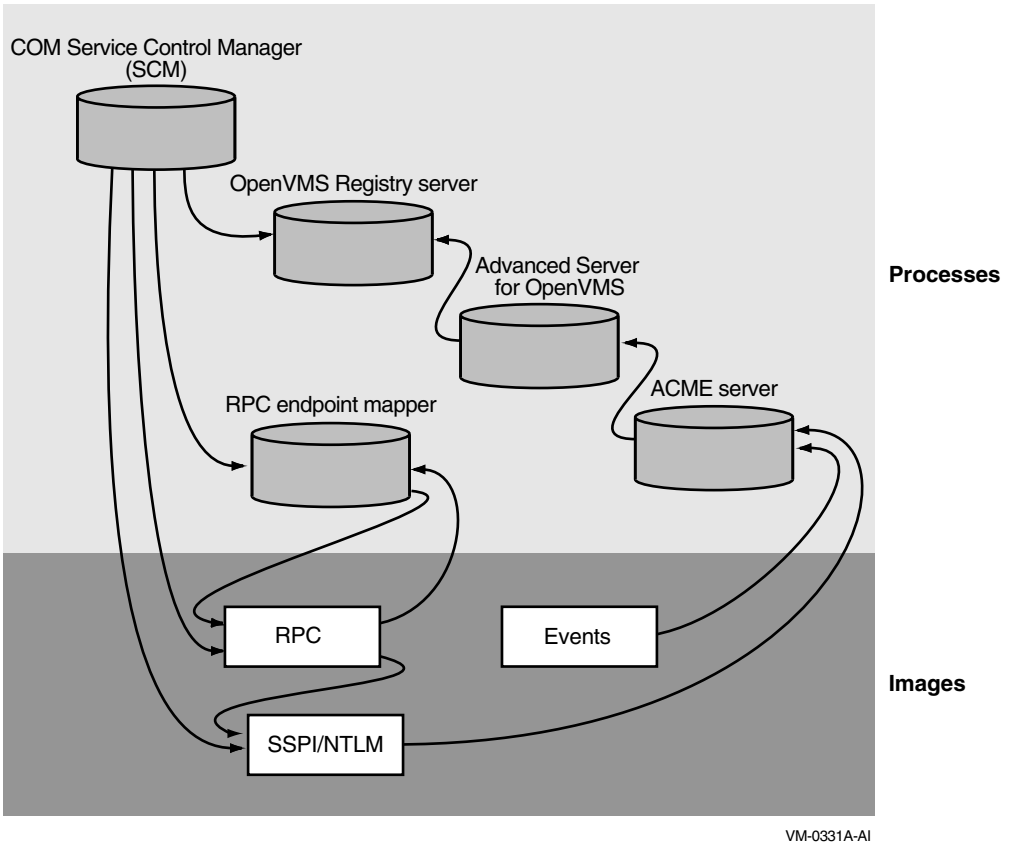


Table 4–1 lists the process names and maps each name to its corresponding server.

Table 4–1 Process Name to Server Name Mapping

Process name	Server name	For more information
DCOM$RPCSS	COM for OpenVMS Service Control Manager (SCM)	Section 4.10.1
REGISTRY_ SERVER	OpenVMS Registry server	Section 4.10.2
PWRK*xxx*	HP Advanced Server for OpenVMS server (multiple processes)	Section 4.10.3
ACME_SERVER	ACME server	Section 4.10.4
DCE$DCED	RPC endpoint mapper	Section 4.10.5

The following sections list and describe the servers and the layers.

4.10.1 COM for OpenVMS Service Control Manager (SCM)

The COM for OpenVMS Service Control Manager enables COM for OpenVMS.

Process name: DCOM$RPCSS

Requires: OpenVMS Registry, OpenVMS (RPC and SSPI/NTLM layers)

Required by: COM applications

Configured by: DCOM$SETUP. See Section 6.2.

Started by: DCOM$SETUP, option 4. See Section 6.2.

Shutdown procedure: DCOM$SETUP, option 5. See Section 6.2.

4.10.2 OpenVMS Registry Server

The OpenVMS Registry server manages the OpenVMS Registry database.

Process name: REGISTRY_SERVER

Requires: None.

Required by: COM for OpenVMS, HP Advanced Server for OpenVMS

Configured by: REG$CONFIG.

Started by: REG$STARTUP. See Section 13.2.1.

Shutdown procedure: SET SERVER REGISTRY_SERVER/EXIT. For more information, see Section 13.3.

4.10.3 HP Advanced Server for OpenVMS Server

The HP Advanced Server for OpenVMS server enables OpenVMS to act as a Windows NTLM domain controller.

The ACME server requires the PWRK$MSV1_0_ACMESHR ACME agent image to talk with the local or remote PWRK$LMSRV process.

Requires: OpenVMS Registry

Required by: ACME server

Configured by: PWRK$CONFIG

Started by: PWRK$STARTUP

Shutdown procedure: PWRK$SHUTDOWN

For more information, see the *HP PATHWORKS for OpenVMS (Advanced Server) Server Migration Guide*.

4.10.4 ACME Server

The ACME server controls the granting of credentials.

Process name: ACME_SERVER

Requires: HP Advanced Server for OpenVMS

Required by: OpenVMS (RPC and SSPI/NTLM layers) and OpenVMS Events

Started:

- Automatically when COM for OpenVMS is started

- You can also start the ACME server manually by entering the following command:

```
$ @SYS$STARTUP:NTA$STARTUP_NT_ACME
```

Shutdown procedures (both are valid):

```
$ SET SERVER ACME /CANCEL /EXIT

$ SET SERVER ACME /DISABLE /CANCEL
$ SET SERVER ACME /ENABLE=(NAME=VMS)
```

For more information, see Section 8.3.2.

4.10.5 RPC Endpoint Mapper

The RPC endpoint mapper controls authentication and security.

Process name: DCE$DCED

Requires: RPC image

Required by: COM for OpenVMS Service Control Manager, RPC image

Started by: OpenVMS

Shutdown procedure: Use the following command procedure:

```
$ @SYS$STARTUP:DCE$RPC_SHUTDOWN.COM
```

For more information, see the *Compaq DCE Installation and Configuration Guide*.

4.10.6 RPC and SSPI/NTLM Layers

The RPC and SSPI/NTLM layers provides remote procedure call and Windows-style authentication on OpenVMS.

Process name: n/a (part of OpenVMS operating system)

Requires: OpenVMS, ACME server

Required by: COM for OpenVMS

Started by: OpenVMS

Shutdown procedure: n/a

4.10.7 OpenVMS Events

The Events layer provides Windows-style event logging on OpenVMS.

Process name: n/a (part of OpenVMS operating system)

Requires: ACME server

Required by: COM for OpenVMS

Started by: OpenVMS

Shutdown procedure: n/a

For more information, see Chapter 15.

4.11 Installing COM for OpenVMS

The COM for OpenVMS installation kit contains a single POLYCENTER Software Installation utility file. The name of the kit is DEC-AXPVMS-DCOM-V0104--1.PCSI. You must install the COM for OpenVMS files on an OpenVMS Alpha system. Please check the prerequisites before installing the kit. See Section 4.2.

To install COM for OpenVMS, invoke the POLYCENTER Software Installation utility using the following command:

```
$ PRODUCT INSTALL DCOM /SOURCE=device:[directory]
```

For *device*:[*directory*], specify the device name and directory location of the kit, respectively.

MIDL compiler license no longer required

The COM for OpenVMS MIDL compiler no longer requires the DCOM-MIDL license.

Example 4–1 shows a sample installation.

Example 4–1 Sample COM for OpenVMS Installation

```
$ product install dcom/source=disk:[directory]

The following product has been selected:
    DEC AXPVMS DCOM V1.4                  Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

DEC AXPVMS DCOM V1.4

 Copyright © 2004 Hewlett-Packard Development Company, L.P. All rights reserved.

Do you want the defaults for all options? [YES]

    The following software is required to run COM for HP OpenVMS

      - HP OpenVMS Alpha V7.3-1 or later
        * Includes DCE RPC and OpenVMS Registry
          (The most recent DCE$LIB_SHR.EXE available is
           required for interoperating with Windows 2000.)
      - HP TCP/IP Services V5.0 or later for HP OpenVMS (or equivalent product)
      - HP Advanced Server V7.3 or later for HP OpenVMS

    Do you want to continue? [YES]

Do you want to review the options? [NO]

Execution phase starting ...

The following product will be installed to destination:
    DEC AXPVMS DCOM V1.4                  DISK$FINALKES:[VMS$COMMON.]

Portion done:
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%

The following product has been installed:
```

4.12 COM for OpenVMS Postinstallation Procedures

After you install the COM for OpenVMS kit, do the following:

1. Verify that the OpenVMS Registry is running. (See Chapter 13.)
2. Verify that the HP Advanced Server for OpenVMS is running. (See Section 4.10.3 for the HP Advanced Server for OpenVMS process names.)
3. Verify that the ACME server is running. (See Section 4.10.4 for the name of this process.)
4. Verify that the RPC daemon is running. (See Section 4.10.5 for the name of the process.)
5. Populate the OpenVMS Registry with the required COM for OpenVMS keys and values using the `DCOM$SETUP` utility, option 3. (See Section 6.2.) You must do this only once in an OpenVMS Cluster.
6. Configure the DCOM$RPCSS account using the `DCOM$SETUP` utility, option 8. (See Section 6.2.1.) You must do this only once in an OpenVMS Cluster.
7. Start COM for OpenVMS using the `DCOM$SETUP` utility, option 4. (See Section 6.2.) You must do this on every node in an OpenVMS Cluster.
8. If you want COM for OpenVMS to start automatically when the system reboots, modify the `DEFINE DCOM$TO_BE_STARTED` line in the `SYLOGICALS.COM` file. (See Section 4.13.1.) You must do this on every node in an OpenVMS Cluster.
9. Configure COM for OpenVMS security. See Chapter 5.

4.13 Starting COM for OpenVMS (COM for OpenVMS Service Control Manager)

Use the following command to start COM for OpenVMS:

```
$ @SYS$STARTUP:DCOM$STARTUP
```

Alternately, you can run `DCOM$SETUP` and choose option 4. (See Section 6.2.)

The COM for OpenVMS Service Control Manager can be in one of the following states: initializing/running or not started. Depending on the COM for OpenVMS Service Control Manager state, you will see one of the following messages:

- If the COM for OpenVMS Service Control Manager is running on this node, the system reports that the process is already active:

  ```
  DCOM Service Control Manager daemon (DCOM$RPCSS) is active [pid=xxxxxxxx]
  ```

  If the COM for OpenVMS Service Control Manager is initializing on this node, the system reports that the process is already active:

  ```
  DCOM Service Control Manager daemon (DCOM$STARTUP-**) is active [pid=xxxxxxxx]
  ```

- If the COM for OpenVMS Service Control Manager is not started on this node, the system starts COM for OpenVMS as follows:

```
The OpenVMS Registry server is already started on this node.

        ***  DCOM system startup procedure  ***

 Starting DCOM Service Control Manager daemon ( "DCOM$STARTUP-**" ) . . .
 After initialization, the daemon will use process name "DCOM$RPCSS" . . .

%RUN-S-PROC_ID, identification of created process is xxxxxxxx

          ***  DCOM startup successful  ***

    ***  DCOM Startup Procedure Complete  ***
```

4.13.1 Starting COM for OpenVMS Automatically after a Reboot

HP recommends that you modify the SYS$MANAGER:SYLOGICALS.COM command file to control COM for OpenVMS startup.

OpenVMS includes a revised SYLOGICALS.TEMPLATE file that includes new startup commands for COM for OpenVMS and related components. Review the "Coordinated Startup" section of this template file and add the appropriate information to your existing startup files.

To have COM for OpenVMS start automatically when the system boots, copy the following line to your SYLOGICALS.COM file, uncomment the line, and make sure it is set to TRUE:

```
$ DEFINE DCOM$TO_BE_STARTED TRUE
```

If you do not set COM for OpenVMS to start automatically when the system boots, you can start COM for OpenVMS using the DCOM$SETUP OpenVMS COM Tools menu, option 4 (see Section 6.2).

4.14 Shutting Down COM for OpenVMS (COM for OpenVMS Service Control Manager)

Use the following command to shut down COM for OpenVMS:

```
$ @SYS$STARTUP:DCOM$SHUTDOWN
```

Alternately, you can run DCOM$SETUP and choose option 5. (See Section 6.2.)

The COM for OpenVMS Service Control Manager can be in one of the following states: stopped, running, or initializing. Depending on the COM for OpenVMS Service Control Manager state, you will see one of the following messages:

- If the COM for OpenVMS Service Control Manager is stopped on this node, the system reports that there is nothing to shut down:

```
        ***  DCOM system shutdown procedure  ***

   There is no active DCOM$RPCSS daemon on this system.

     ***  DCOM Shutdown Procedure Complete  ***
```

- If the COM for OpenVMS Service Control Manager is running on this node, the system shuts down the process as follows:

```
        ***  DCOM system shutdown procedure  ***

************************** Warning ********************************
***  Stopping the DCOM Service Control Manager daemon (DCOM$RPCSS)
***  Active DCOM applications will no longer be operational.

Do you want to proceed with this operation (YES/NO/?) [N]?
```

Enter Y to continue with the shutdown procedure.

Note

For information about suppressing this confirmation step, see Section 4.14.1.

The system displays the following messages:

```
Terminating DCOM Service Control Manager daemon (DCOM$RPCSS) . . .

     ***  DCOM shutdown successful  ***

   ***  DCOM Shutdown Procedure Complete  ***
```

- If the COM for OpenVMS Service Control Manager is initializing on this node, the system shuts down the process as follows:

```
       ***  DCOM system shutdown procedure  ***

************************** Warning ******************************
***  Stopping the DCOM Service Control Manager daemon (DCOM$RPCSS)
***  Active DCOM applications will no longer be operational.

Do you want to proceed with this operation (YES/NO/?) [N]?
```

Enter Y to continue with the shutdown procedure. The system displays the following messages:

```
Terminating DCOM Service Control Manager daemon (DCOM$STARTUP-**) . . .

     ***  DCOM shutdown successful  ***

   ***  DCOM Shutdown Procedure Complete  ***
```

4.14.1 Suppressing the DCOM$SHUTDOWN Confirmation Request

You can suppress the DCOM$SHUTDOWN command confirmation request by specifying the NOCONFIRM parameter. Use the following command:

```
$ @SYS$STARTUP:DCOM$SHUTDOWN NOCONFIRM
```

The system displays the following shutdown messages without prompting you to confirm the shutdown:

```
      ***  DCOM system shutdown procedure  ***

Terminating DCOM Service Control Manager daemon (DCOM$RPCSS) . . .

     ***  DCOM shutdown successful  ***

  ***  DCOM Shutdown Procedure Complete  ***
```

5
COM for OpenVMS Security

COM V1.1-A and higher for OpenVMS supports NTLM (NT LAN Manager) authentication for controlling access to COM objects.

Processes that execute client and server applications must obtain Windows credentials in order to be authenticated. Processes created automatically by DCOM$RPCSS to execute server applications obtain Windows credentials based on the Registry settings for the server being launched. Interactive processes that are used to execute client and server applications must obtain Windows credentials by running the NTA$LOGON utility (see Section 8.2).

This chapter applies to COM for OpenVMS in authenticated mode. See Appendix E for information about running COM for OpenVMS in an unauthenticated environment.

This chapter discusses the following topics:

- How to configure an OpenVMS system for NTLM authentication
- How to acquire Windows credentials
- The way security affects COM applications
- The way your domain configuration affects COM applications
- The Application Server run-time environment

5.1 System Configuration

NTLM authentication on OpenVMS is implemented in three major components of the operating system (see Section 4.10).

- ACME server — controls the granting of credentials
- RPC and SSPI — provide remote procedure calls and Windows-style authentication
- Advanced Server for OpenVMS — maintains Windows accounts and provides mapping of Windows accounts to OpenVMS accounts

The ACME server, RPC, and SSPI are installed as part of the OpenVMS operating system and require no special configuration. Advanced Server for OpenVMS must be installed as a layered product and must be configured to support NTLM authentication for COM applications (see Section 4.4).

After installing Advanced Server for OpenVMS, you must create Windows domain accounts that will be used to execute COM applications. You must also map the Windows domain accounts to OpenVMS accounts.

The Advanced Server ADMINISTER utility is used to create Windows domain accounts. For example, to create the Windows domain account NTUSER1, use the following command:

```
$ ADMINISTER ADD USER NTUSER1 /PASSWORD="pppppp" /FLAG=NOPWDEXPIRED
```

The password is case sensitive, so it is enclosed in quotation marks in order to maintain case. A password without quotation marks is converted to uppercase. By default, Windows domain accounts are created with the password pre-expired, thus forcing the user to change the password at the first login. The NOPWDEXPIRED flag overrides this default.

A hostmap entry defines the association between a Windows user account and a local OpenVMS user account. When OpenVMS authenticates a Windows user, OpenVMS uses the hostmap entry to map the OpenVMS user account to the Windows user account and build the local OpenVMS user profile and the Windows NT user profile. If no hostmap entry exists, OpenVMS uses the Windows user account name as the local OpenVMS user account name.

Use the HP Advanced Server for OpenVMS ADMINISTER utility to define hostmap information. For example, to map the Windows domain account NTUSER1 to the OpenVMS account VMSUSER1, use the following command:

```
$ ADMINISTER ADD HOSTMAP NTUSER1 VMSUSER1
```

If the OpenVMS account does not already exist, you must create the account using the OpenVMS Authorize utility (AUTHORIZE). The OpenVMS account must have the EXTAUTH flag set, or the IGNORE_EXTAUTH flag (bit 11, %X0800) must be set in the SECURITY_POLICY SYSGEN parameter (see Section 5.1). This policy allows the OpenVMS system manager to control which OpenVMS user accounts can be used with Windows authentication. For example, to set the EXTAUTH flag for an OpenVMS account VMSUSER1, use the following command. For example:

```
$ AUTHORIZE == "$AUTHORIZE"
$ AUTHORIZE MODIFY VMSUSER1 /FLAG=EXTAUTH
```

5.1.1 LOGINOUT.EXE Use of External Authentication

The EXTAUTH flag also directs `LOGINOUT.EXE` to use external authentication to authenticate an OpenVMS user during the login process (that is, local, dialup, remote, interactive, and network logins). When you set the EXTAUTH flag, `LOGINOUT.EXE` uses external authentication, not the OpenVMS SYSUAF.DAT record, to verify the user name and password.

LOGINOUT external authentication always requires that you set the EXTAUTH flag in the SYSUAF account record. Unlike `NTA$LOGON` and authenticated RPC, you cannot override this requirement using the IGNORE_EXTAUTH flag.

5.1.2 DCE Integrated Login Restriction

A site cannot use both external authentication and the older LGI-callout feature on the same system. If you have an LGI-callout image installed, external authentication is disabled for login purposes. Because DCE integrated login uses the LGI-callout mechanism, OpenVMS does not allow logins using Windows-based external authentication if DCE integrated login is enabled.

5.2 Cross-Domain Configuration

You can run a COM application on a system in one domain and have the application authenticated by a system in a second domain.

To configure authentication across Windows domains, you must do the following:

1. Set up trust relationships between domains.

   For more information, see the *HP Advanced Server for OpenVMS Server Administrator's Guide*.

2. Set up the HostMapDomains parameter on HP Advanced Server for OpenVMS domains (see Example 5–1).

   For more information, see the *HP Advanced Server for OpenVMS Server Administrator's Guide*.

3. Set up account hostmap entries between the Windows user account and a local OpenVMS user account.

Example 5–1 shows how you can set up the HostMapDomains parameter. In this example, there are two domains: `DOMAIN_1` and `DOMAIN_2`. Domain `DOMAIN_2` is running HP Advanced Server for OpenVMS; domain `DOMAIN_1` is a Windows domain. The commands in Example 5–1 introduce `DOMAIN_2` to `DOMAIN_1`.

Example 5–1 Sample: Setting Up HostMapDomains

```
SYSJANE$ show sym regutl
  REGUTL == "$SYS$SYSTEM:PWRK$REGUTL.EXE"
SYSJANE$ regutl
REGUTL> SET PARAM /CREATE VMSSERVER HOSTMAPDOMAINS DOMAIN_1
REGUTL> SHOW VALUE * HOSTMAPDOMAINS
Key: SYSTEM\CurrentControlSet\Services\AdvancedServer\UserServiceParameters
Value: HostmapDomains
Type: String
Current Data: DOMAIN_1
```

5.3 Acquiring Windows Credentials

After the Windows domain account and the OpenVMS account have been set up as described in Section 5.1, you can log in to the OpenVMS account using the usual OpenVMS login procedures. You can then acquire Windows credentials using the `NTA$LOGON` utility. For example:

```
$ NTA$LOGON == "$NTA$LOGON"
$ NTA$LOGON NTUSER1 "pppppp"
```

In this format, *pppppp* is the password you specified when you created the Windows domain account. The password is enclosed in quotation marks to preserve case. A password without quotation marks is converted to lowercase. If the user name or password is not specified on the command line, the program prompts the user for the required input (see Section 8.2).

To acquire Windows credentials using `NTA$LOGON`, you must be logged in to the OpenVMS account to which the Advanced Server account is hostmapped. If not are not logged in, you must have the IMPERSONATE privilege and use the NTA$LOGON /OVERRIDE_MAPPING option. For example:

```
$ NTA$LOGON == "$NTA$LOGON"
$ NTA$LOGON /OVERRIDE_MAPPING NTUSER2 "pppppp"
```

To determine whether a process has Windows credentials, use the NTA$LOGON utility with the /LIST switch. For example:

```
$ NTA$LOGON == "$NTA$LOGON"
$ NTA$LOGON /LIST
```

5.4 Application Security

The COM security model allows the creation of secure distributed applications. COM security can be enabled by using settings in the OpenVMS Registry and by using COM security APIs and interfaces. There are two primary areas of security that can be applied to COM applications: launch security and activation security.

Launch security and activation security have system default settings; application-specific settings override these defaults. The settings are stored in the Registry and are maintained by using the DCOMCNFG utility on Windows and by using the DCOMCNFG option of DCOM$SETUP.COM on OpenVMS. The COM API CoInitializeSecurityEx can be used from within an application to enhance or override the Registry settings.

5.4.1 Launch Security

Launch security determines which Windows domain accounts can be used to create, or "launch" server processes. The launch security settings are referenced when a COM request is received on a system that will result in the launching of a server process to satisfy the request. These settings can explicitly or implicitly allow or disallow a user request to launch a server. The DCOM$RPCSS process authenticates the incoming request to determine the identity of the client. If DCOM$RPCSS determines that it needs to launch a server process to satisfy the request, DCOM$RPCSS allows or disallows the launching of the server based on the identity of the client and the launch security settings.

5.4.2 Activation Security

Activation security determines which Windows domain accounts can be used to execute method calls in server applications. The activation security settings are referenced when a COM request is received on a system for a method call in an existing server process. The server process authenticates the incoming request to determine the identity of the client. The server process allows or disallows the execution of the method call based on the identity of the client and the activation security settings.

5.4.3 Server Process Identity

A server process created by DCOM$RPCSS on OpenVMS is a detached process that has an OpenVMS identity and follows all the OpenVMS security rules for a detached process. In addition, it has a network identity that is used to enforce the COM security model (see Section 5.5).

COM servers create separate server threads to execute each client request. These server threads have their own OpenVMS identity and network identity, based on the identity of the client. When a server thread is executing a request on behalf of a client, it is the thread's identities, not the process' identities, that are used to enforce security.

5.4.4 Domain Issues

Two systems running COM client and server applications can exist in one of three possible domain configurations:

- Systems are in the same domain
- Systems are in separate domains with trusts established between the two domains
- Systems are in separate domains without trusts, or systems are not in a domain

The ability for servers and `DCOM$RPCSS` to authenticate client requests are affected by the domain configurations. When both systems are in the same domain or when the systems are in separate but trusted domains there is no problem authenticating. The trusted domain configuration is a bit more complex and requires that the trusts and mappings be configured correctly but once configured, there is no trouble authenticating (see Section 5.2).

Systems in separate, nontrusted domains or systems not in any domain cannot be authenticated using the normal mechanisms. To run authenticated COM applications between such systems, you must pass authentication information (user name and password) from the client to the server. COM provides this capability in the CoCreateInstanceEx API. The pServerInfo parameter of the CoCreateInstanceEx API allows you to specify a user name and password to be used for authentication on the remote server system. The user name and password are part of the COAUTHIDENTITY structure, within the COAUTHINFO structure within the COSERVERINFO structure, that is passed as the pServerInfo parameter to CoCreateInstanceEx.

Section C.3 shows how you can authenticate a remote client that is neither in the server's domain nor in a domain that has a trust with the server's domain.

The current NTLM security implementation on OpenVMS does not support this feature for COM client applications on OpenVMS. This feature is supported for COM clients on Windows that communicate with COM servers on OpenVMS. To run COM client applications on OpenVMS where the server is not in the same domain or in a trusted domain, you must disable authentication for the application, as described in Section 5.4.5.

5.4.5 Disabling Authentication

Under certain conditions, you may want to disable authentication between a client and server applications. This feature disables many of the security features of COM and of the operating system and should not be used in an environment where security is required. There are two ways to disable authentication for COM applications:

- Use `DCOMCNFG` to change the default authentication level to **None** on both systems.
- Add a call to `CoInitializeSecurity` in both the client and server applications and set the `dwAuthnLevel` parameter to `RPC_C_AUTHN_LEVEL_NONE`.

The server must be configured to run with a specific NTLM account identity. Since the client will not be authenticated, there is no way for the server to run with a client's identity. To configure a server to run with a specific NTLM identity, use `DCOMCNFG` and change the application properties to select the NTLM account.

5.4.6 Access Denied Problems (80070005)

The most common security error a COM application will encounter is access denied (error status value 80070005). The following is a list of the most common causes of this error:

- Client process on OpenVMS does not have Windows credentials. Run NTA$LOGON to acquire Windows credentials.
- Application-specific launch or access permissions do not allow access. Check the application settings using DCOMCNFG (see Section 6.3.1).
- System default launch or access permissions do not allow access. Check the system defaults using DCOMCNFG (see Section 6.3.6).
- CoInitializeSecurityEx API call is incorrect. Verify that the client and server security calls are valid.
- Server process or thread does not have permission to perform a particular operation. Verify that the OpenVMS identity being used to execute a client request has the necessary privileges to perform the operation.
- Server process does not have access to the server images. Verify that the OpenVMS identity used to launch a server process has read and execute permissions for the server image and any dynamically loaded images (.EXE files).
- Advanced Server hostmap entry problems. In order for NTLM authentication to work correctly, Windows domain accounts must be mapped to OpenVMS accounts (see Section 5.1). To verify the mapping of Windows domain accounts to OpenVMS accounts, use the ADMINISTER command:

  ```
  $ ADMINISTER SHOW HOSTMAP
  ```

  In a multiple-domain environment with trusts established between domains, cross-domain mappings must be created (see Section 5.2). Under certain conditions, the Advanced Server is unable to verify a user account associated with a hostmap entry. Any attempt to display the hostmap entry of a user name that can not be verified will result in the user name being displayed using its eight-digit hexadecimal internal representation (for example, "DOMAINNAME\000003fd"). If this happens, verify that the Advanced Server is running on each machine. You should also verify the trusts between domains using the following ADMINISTER command:

  ```
  $ ADMINISTER SHOW TRUST
  ```

  If the trusts are valid and the hostmap entries are still displayed with the numeric format, you should shut down and restart the Advanced Server.

5.5 Server Run-Time Environment

When DCOM$RPCSS launches a server in response to a client request for a COM object, DCOM$RPCSS creates a detached process and executes either the server image or server command file in the context of the detached process. The image or command file that is executed is determined by the value of the Registry key HKEY_CLASSES_ROOT\CLSID\{*iid*}\LocalServer32, where *iid* is the unique identifier of the COM object.

The run-time environment of the detached process is as follows:

- Default directory

The default directory of the detached process is the same as the default directory of DCOM$RPCSS. This is determined by the default directory of the process that executed the DCOM$STARTUP command file. If DCOM$STARTUP is executed by the system startup procedure, then the default directory will be SYS$SYSTEM.

- Windows identity

  Depends on the application identity setting. This setting is made using DCOMCNFG.

  If the application identity is set to "launching user" (the default), then the Windows identity of the detached process is the same as the Windows identity of the client.

  If the application identity is set to a specific NTLM account then the Windows identity of the detached process is that of the NTLM account.

- OpenVMS user name

  The OpenVMS user name of the detached process is the user name that is mapped to the Windows identity of the detached process. The mapping of OpenVMS user name to Windows identity is established using the Advanced Server ADMINISTER utility (see Section 5.1).

- OpenVMS privileges

  Depends on the application identity setting. This setting is made using DCOMCNFG.

  If the application identity is set to "launching user" (the default), then the privileges depend on the location of the client. If the client is running on the same system as the server, then the privileges of the detached process will match the privileges of the client. If the client is running on a different system from the server, then the privileges of the detached process will be the default privileges of the OpenVMS user name account.

  If the application identity is set to a specific NTLM account then the privileges of the detached process will be the default privileges of the OpenVMS user name account.

- Process logicals

  SYS$INPUT and SYS$OUTPUT are defined to the disk device where the server image or command file is located. For example, if the server image is DKA0:[TEST]CMPNT.EXE, then SYS$INPUT and SYS$OUTPUT are defined to DKA0:. SYS$SCRATCH is not defined.

  If these environment settings are not sufficient for the successful execution of your server, then you should explicitly define the environment settings you need. One way to easily set up an environment for your server is to create a command file to run your server, and register the command file, instead of the executable image, as the file to be executed when the server is launched. You can define the environment in the command file prior to executing the server image. For example, if you build SAMPLE1 in the directory DKA0:[SAMPLE1] and register it using the BUILD_SAMPLE1 command file, then the server image will be named DKA0:[SAMPLE1]CMPNT.EXE. The Registry key HKEY_CLASSES_ROOT\CLSID\{0C092C21-882C-A6BB-0080C7B2D682}\LocalServer32 will have a value of DKA0:[SAMPLE1]CMPNT.EXE. You can change the value of that key to DKA0:[SAMPLE1]RUN_CMPNT.COM and create the command file, as follows:

```
$! RUN_CMPNT.COM
$! Command file to run SAMPLE1
$ set default DKA0:[SAMPLE1]
$ define sys$output DKA0:[SAMPLE1]SAMPLE1.LOG
$ ! Other definitions as needed
$  RUN CMPNT.EXE
$ exit
```

When DCOM$RPCSS receives a request for SAMPLE1 and launches a server, the server executes this command file in the detached process.

6

COM for OpenVMS Utilities for Application Development and Deployment

This chapter describes how to configure your OpenVMS system (and, optionally, your Windows system) to develop and deploy COM applications. It describes the following COM for OpenVMS utilities:

- The DCOM$SETUP utility, which helps a system manager configure the COM for OpenVMS system environment.
- The DCOM$CNFG utility, which helps an application developer configure and examine COM applications.
- The DCOM$REGSVR32 utility, which allows an application developer to register and unregister in-process server applications.

This chapter also includes information about configuring OpenVMS and Windows systems to interoperate.

Before you begin

Before you configure COM for OpenVMS on your OpenVMS system, you must install and configure required components and install COM for OpenVMS. See Chapter 4 for information about these steps.

6.1 DCOM$SETUP Utility

DCOM$SETUP is a collection of tools to help a system manager configure the COM for OpenVMS system environment.

DCOM$SETUP Conventions and Requirements

- For Yes/No questions, you can enter any one of the following:
  - YES or NO
  - Y or N
  - Return (to accept the default value)
- Some DCOM$SETUP options require system manager privileges and OpenVMS Registry access.

6.2 Running DCOM$SETUP

To run DCOM$SETUP, enter @SYS$STARTUP:DCOM$SETUP at the OpenVMS system prompt.

The system displays the OpenVMS COM Tools menu.

Figure 6–1 DCOM$SETUP OpenVMS COM Tools Menu

```
------------------------------------------------------
                        OpenVMS COM Tools

          1) DCOMCNFG, COM Configuration Properties
          2) GUIDGEN, Globally Unique Identifier Generator
          3) Populate the Registry database for COM
          4) Start the COM server
          5) Stop the COM server
          6) Register a COM application
          7) Create the DCOM$GUEST account and directory
          8) Configure the DCOM$RPCSS accounts

          H) Help
          E) Exit

Please enter your choice:
------------------------------------------------------
```

To choose an option, enter the option number. The options are as follows:

- 1) DCOMCNFG, COM Configuration Properties

  Use to query information and manipulate properties of COM for OpenVMS applications. For more information, see Section 6.3.

- 2) GUIDGEN, Globally Unique Identifier Generator

  Generate CLSIDs (class IDs) (or GUIDs [globally unique identifiers]) in various formats (for example, the OpenVMS Registry or Windows Registry format). The CLSID tags each application with a unique identifier.

  This version of DCOM$SETUP generates GUIDs in OpenVMS Registry and Windows Registry formats only. For a discussion of other formats, see Section 7.1.

- 3) Populate the Registry database for COM

  Set up the OpenVMS Registry database. COM for OpenVMS requires that specific keys and values be added to the OpenVMS Registry database. You must have both write access to the OpenVMS Registry and Windows Administrator privileges.

- 4) Start the COM server

  Start the COM for OpenVMS Server Control Manager server (DCOM$RPCSS). DCOM$SETUP calls the SYS$STARTUP:DCOM$STARTUP procedure to start the server. For more information, see Section 6.2.2.

- 5) Stop the COM server

  Shut down the COM for OpenVMS Service Control Manager server (DCOM$RPCSS). DCOM$SETUP calls the SYS$STARTUP:DCOM$SHUTDOWN procedure to stop the server. For more information, see Section 6.2.2.

- 6) Register a COM application

Register a COM for OpenVMS server application. You can register the following types of servers:

— In-process server

When you register an in-process server, the system prompts you for the server's location.

— Local server or out-of-process server

When you register a local server or out-of-process server, the system prompts you for the following information:

+ Full path information (location of the server)

This is a required value. Use the following syntax: *device*::[*directory*]*file-name.ext*

+ Application title

This is an optional value. If you do not supply a title, the system uses a default title.

+ CLSID (GUID)

This is a required value. If the server does not have a CLSID, the system generates one automatically. For more information about CLSIDs and LocalServer32, see Section 7.5.1.

After you complete the registration process, the system generates the following files:

1. A Windows Registry file (*server-name*.REG_NT) that you can use to register the application on a Windows system.
2. An OpenVMS command procedure (*server-name*.REG_VMS) that you can use to register the server on an OpenVMS system.

When you use these files on other systems, you must modify the path statement to point to the server's current location. For more information, see Section 6.2.3.

- 7) Create the DCOM$GUEST account and directory

  You must create the DCOM$GUEST account before you can use COM for OpenVMS without NTLM authentication.

- 8) Configure the DCOM$RPCSS accounts

  Configure and create the DCOM$RPCSS HP Advanced Server for OpenVMS user and SYSUAF accounts. The COM for OpenVMS Service Control Manager (DCOM$RPCSS) requires these accounts for authentication. For more information, see Section 6.2.1.

- H) Help

  Display help about each menu option.

- E) Exit

  Exit the menu.

6.2.1 Creating and Configuring DCOM$RPCSS Accounts

To display these functions, choose option 8 from the OpenVMS COM Tools menu. The system displays the following:

```
------------------------------------------------------
Configure the COM for OpenVMS Service Control Manager (DCOM$RPCSS) accounts

   1) Create the DCOM$RPCSS account in both the SYSUAF database and the
      Advanced Server for OpenVMS SAM database. The password you specify
      for the new DCOM$RPCSS user is stored in a protected file.

   2) Update the DCOM$RPCSS user password in the COM for OpenVMS Service
      Control Manager password file.

   E) Exit

Please enter your choice:
------------------------------------------------------
```

Enter one of the following:

- `1) Create the DCOM$RPCSS account . . .`

  This option creates the `DCOM$RPCSS` account in both the SYSUAF database and the HP Advanced Server for OpenVMS SAM database.

  The password you specify for the DCOM$RPCSS user is stored in a protected file that the COM for OpenVMS Service Control Manager uses to log into the NTLM network and obtain a Windows identity.

  Note

  The system creates this account in the HP Advanced Server for OpenVMS database with a password that will not expire. To change this behavior (that is, modify the account so that the password expires according to the HP Advanced Server for OpenVMS User Policy), use the following procedure:

  1. Run the HP Advanced Server for OpenVMS ADMIN utility.
  2. Log into the Administrator account.
  3. Issue the following ADMIN command:

     ```
     ADMIN> MODIFY USER DCOM$RPCSS/FLAG=NODISPWDEXP
     ```

  To determine the maximum password age in the HP Advanced Server for OpenVMS User Policy, enter the following ADMIN command:

  ```
  ADMIN> SHOW ACCOUNT POLICY
  ```

  If you change the HP Advanced Server for OpenVMS password of the DCOM$RPCSS account, you must update the password in the COM for OpenVMS Service Control Manager password file. (See option 2 [Update the DCOM$RPCSS user password].)

  Use the following procedure:

  1. Enter `1`.

The system displays the following:

```
        To create a new account, you must be logged on to an existing
        Advanced Server for OpenVMS account that is capable of adding
        new users.

Enter Y[ES] to log on to this account:
```

You must belong to the PATHWORKS administrator group to create this account.

2. Enter `Y`.

   The system prompts you to log on. The password is not displayed as you enter it.

```
Enter username: JOSEPHM
Password:
Confirm password:
```

   The system prompts you to enter a new password, and then asks you to confirm the password. The password is not displayed as you enter it.

```
        Enter the new DCOM$RPCSS password.

Enter password:
Confirm password:
```

   The system uses this password for both the SYSUAF account (DCOM$RPCSS) and the PATHWORKS user account (DCOM$RPCSS). The system stores this password in the COM for OpenVMS Service Control Manager password file.

   The system displays the following account creation information:

```
%PWRK-S-USERADD, user "DCOM$RPCSS" added to domain "DCOM1_DOMAIN"

Username: DCOM$RPCSS                        Owner:  COM
Account:                                    UIC:    [37776,1] ([DCOM$RPCSS])
CLI:      DCL                               Tables: DCLTABLES
Default:  SYS$SYSDEVICE:[DCOM$RPCSS]
LGICMD:
Flags:  ExtAuth
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
No access restrictions
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:           (none)    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive),           (none) (non-interactive)
Maxjobs:         0  Fillm:       100  Bytlm:        64000
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:       150  JTquota:       4096
Prclm:           8  DIOlm:       150  WSdef:         1024
Prio:            4  ASTlm:       250  WSquo:         4000
Queprio:         4  TQElm:        10  WSextent:      8000
CPU:        (none)  Enqlm:      2000  Pgflquo:     130000
Authorized Privileges:
  NETMBX       TMPMBX
Default Privileges:
  NETMBX       TMPMBX
%PWRK-S-HOSTMAPADD, user "DCOM$RPCSS" mapped to host user "DCOM$RPCSS"

Press RETURN to continue:
```

- 2) Update the DCOM$RPCSS user password . . .

If you change the DCOM$RPCSS user password in the HP Advanced Server for OpenVMS SAM database, you must also update the password in the COM for OpenVMS Service Control Manager password file.

Use the following procedure:

1. Enter 2.

   The system displays the following:

```
        Enter the new DCOM$RPCSS password.

Enter password:
Confirm password:
```

2. Enter the new password and confirm the password.

- E) Exit

  Exit the menu.

6.2.2 Starting and Stopping the COM Server (DCOM$RPCSS Process)

COM for OpenVMS requires that the COM server process (DCOM$RPCSS) always be running. The DCOM$RPCSS process on OpenVMS provides the same functions for the COM run-time environment that the RPCSS process provides on Microsoft Windows, including the following:

- Build and maintain the list of server objects running on the system.
- Build and maintain a cache of known applications as defined in the registry. This cache improves COM performance.
- Start a server as a detached process whenever a client requests a connection to a server object that is not currently running.
- Communicate with the RPCSS process on remote Windows systems or the DCOM$RPCSS process on OpenVMS systems to locate or start remote server objects.

To start DCOM$RPCSS, either use DCOM$SETUP option 4 (“Start”) (see Section 6.2) or call the COM for OpenVMS startup procedure directly from SYS$STARTUP:DCOM$STARTUP. See Section 4.13 for information on starting COM for OpenVMS.

To stop DCOM$RPCSS on your system, either use the DCOM$SETUP option 5 (“Stop”) (see Section 6.2) or call the COM for OpenVMS shutdown procedure directly from SYS$STARTUP:DCOM$SHUTDOWN. See Section 4.14 for information on shutting down COM for OpenVMS.

6.2.3 Registering an Application

The following example shows how to register the COM for OpenVMS “Simple” application included on the COM for OpenVMS kit. You can use the resulting Windows file to register the server on a Windows system as long as the application is available on your Windows system.

To build the “Simple” application on a Windows system, see and execute the instructions in the README-SIMPLE.TXT file in DCOM$EXAMPLES:[SIMPLE].

Note

You must build and compile the application before you can register it. For complete details, see the step-by-step example in

DCOM$EXAMPLES:[SIMPLE] included in the COM for OpenVMS kit.

Use the following procedure:

1. From the DCOM$SETUP menu, enter 6 or REGISTER.
2. Answer the questions as follows:

Note

The "Simple" application already has a CLSID.

Example 6–1 Sample "Simple" Application Registration on OpenVMS

```
Enter server type (1. In-Proc  2. Out-Proc): 2 Return
Enter Local Path (device:[directory]filename.ext): DKA0:[SMITH]SSERVER.EXE Return
Enter Application Name (<RETURN> to assign default): COM Simple Server Return
Does the server have a CLSid {GUID} (Yes/No) [N]: Y Return
Enter the CLSid (i.e. {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}:
   {5e9ddec7-5767-11cf-beab-00aa006c3606} Return

Verify Application Information:

         Application Name: COM SIMPLE SERVER
         Local Path: DKA0:[SMITH]SSERVER.EXE
         Application ID: {5E9DDEC7-5767-11CF-BEAB-00AA006C3606}

Is the information correct (Yes/No) [Y]: Return
Register application (Yes/No)? [Y]: Return

SETUP-I-NEWFILES, The following files have been created:

      DKA0:[SMITH]SSERVER.REG_NT
      DKA0:[SMITH]SSERVER.REG_VMS

SETUP-I-SRVIN, Server has been registered
Press RETURN to continue: Return
```

To register the "Simple" application on a Windows system, use the following procedure:

1. Copy all the files in the DCOM$EXAMPLES:[SIMPLE] directory to your Windows system.
2. Rename SSERVER.REG_NT to SSERVER.REG.
3. Edit the file to point to the local server path.

   For example, replace DEVICE:\SSERVER with C:\SSERVER.
4. Run the Install.bat program to add the necessary keys to the Windows registry.

Example 6–2 shows the contents of SSERVER.REG_NT.

Example 6–2 Contents of SSERVER.REG_NT

```
REGEDIT
HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\ = DCOM server application SSERVER
HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LaunchPermission = Y
HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LocalServer32 = DEVICE:\SSERVER
```

To reregister the “Simple” application on an OpenVMS system, enter the following command at the system prompt:

```
$ @SSERVER.REG_VMS
```

Example 6–3 shows the contents of the SSERVER.REG_VMS command procedure:

Example 6–3 Contents of SSERVER.REG_VMS

```
$ Set noon
$ regcp :=  $regcp
$ crekey := $regcp create key
$ creval := $regcp create value
$ modval := $regcp modify value
$ lisval := $regcp list value
$ crekey HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}
$ creval HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606} -
             /data="DCOM server application SSERVER" /type=sz
$ creval HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}/name="AppID" -
             /data="{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}" /type=sz
$ crekey HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LaunchPermission
$ creval HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LaunchPermission -
             /data="Y" /type=sz
$ crekey HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LocalServer32
$ creval HKEY_CLASSES_ROOT\CLSID\{5E9DDEC7-5767-11CF-BEAB-00AA006C3606}\LocalServer32 -
             /data="DKA0::[SMITH]SSERVER.EXE" /type=sz
$
```

6.3 Running DCOM$CNFG

DCOM$CNFG is a utility to help COM developers configure and manage COM for OpenVMS applications on OpenVMS. Use the DCOM$CNFG utility to query information and manipulate properties of COM for OpenVMS applications.

To use the DCOM$CNFG utility, choose option 1 from the DCOM$SETUP menu.

Note

Before running the DCOM$CNFG utility, you must:

- Have OpenVMS Registry **Read** access to read application properties, and **Write** access to modify application properties.
- Ensure that the ACME server is running on the current system. The ACME server must be running to view and change application security properties. For more information, see Table 4–1.
- Acquire Windows security credentials before you can change an application identity. For more information, see Section 8.2.

The system displays the DCOM$CNFG Main menu.

Figure 6–2 DCOM$CNFG Main Menu

```
------------------------------------------------------
               DCOM$CNFG Main

   1 - Applications List
   2 - System-wide Default Properties
   3 - System-wide Default Security

   (E to Exit)
   (H for Help)

   Enter <CTRL-Z> or 'E' to return to the previous menu at any time

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Applications List`

  Lists all applications registered on this machine. For more information about this option, see Section 6.3.1.

- `2 - System-wide Default Properties`

  Allows you to set systemwide machine properties. For more information about this option, see Section 6.3.5.

- `3 - System-wide Default Security`

  Allows you to set systemwide security parameters. For more information about this option, see Section 6.3.6.

6.3.1 The DCOM$CNFG Application List Submenu

To display this submenu, from the `DCOM$CNFG` Main menu, choose option 1.

The system displays the Applications List submenu.

Figure 6–3 Applications List Submenu

```
------------------------------------------------------
               Applications List

   Index    Name
    1      Inside COM, Chapter 11 Example
    2      application 2
    3      application 3
    .      ...
    .      ...
    .      ...

  (E to Exit to previous menu)
  (H for Help)

  Please enter Index number to select an Application:
------------------------------------------------------
```

Enter a number to select an application. You can then view or configure its properties.

This option displays the Application Properties submenu.

Note

The system stores the Application Properties (Location, Security, and Identity) (see Figure 6–4) in a special key in the OpenVMS Registry that is associated with each application. You cannot change the Application Properties until you create this special key using the DCOM$CNFG utility. The DCOM$CNFG utility creates this special key when the utility discovers a newly registered application. In this case, the DCOM$CNFG user must have acquired Windows security credentials for an account that is a member of the Administrator group. Otherwise, the key will not be created. For more information about acquiring Windows credentials, see Section 8.2).

Use the following procedure to manage the Application Properties:

1. Register the application.
2. Do either of the following:
   - Acquire Windows security credentials for an account that is a member of the Administrator group and then run DCOM$CNFG.
   - Have a system administrator with the appropriate credentials run DCOM$CNFG.
3. Run DCOM$CNFG from your own account to manage the properties.

Figure 6–4 Application Properties Submenu

```
------------------------------------------------------------------
              Application Properties

   General Properties of this DCOM Application

   Application name:  Inside COM, Chapter 11 Example
   Application id:    {0C092C2C-882C-11CF-A6BB-0080C7B2D682}
   Application type:  local server
   Local path:        DISK1:[SMITH.DISPATCH_SAMPLE1]CMPNT.EXE
   Type Library:      {D3011EE1-B997-11CF-A6BB-0080C7B2D682}
   version: 1.0  DISK1:[SMITH.DISPATCH_SAMPLE1]Server.tlb

   1 - Location    Machine to run application
   2 - Security    Security permissions for application
   3 - Identity    User account to use to run application

   (E to Exit to previous menu)
   (H for Help)

   Please enter Application Property you wish to change:
------------------------------------------------------------------
```

If the system cannot find the type library file or if the type library is unaccessible, the system displays an error message next to the type library file name.

The options are as follows:

- `1 - Location: Machine to run application`

  This option allows you to set or change the machine on which the COM application will run.

The system displays the Application Location submenu.

Figure 6–5 Application Location Submenu

```
------------------------------------------------------
               Application Location

   The following settings allow DCOM to locate the correct computer
   for this application. If more than one machine is selected then
   DCOM uses the first available one. Client applications may override
   these selections.

   Application name:  Inside COM, Chapter 11 Example

   1 - Run application on this computer (Yes/No)
          Current value: Yes

   2 - Run application on another computer
          Current value: Currently Disabled

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

— `1 - Run application on this computer`

Indicates whether the application will be run on the local computer. Select the option to change the current value.

— `2 - Run application on another computer`

Indicates that the application will be run on the specified computer. Select the option and enter one of the following:

+ A valid system name to change the current value.

+ A hyphen (-) to disable the value. This sets the field to "Currently Disabled."

- `2 - Security: Security permissions for application`

  This option allows you to set the following security properties:

  — Access permission: allow or deny access to users or groups to access this application.

  — Launch permission: allow or deny access to users or groups to run this application.

  — Configuration permission: identify users or groups who have read, write, or special access to the OpenVMS Registry area that contains information about the application.

- `3 - Identity: User account to use to run application`

  This option allows you to run the application server using the security context of the specified user account.

  The system displays the Application Identity submenu. See Section 6.3.4.

The system uses the systemwide default security values unless you specify a different setting.

The system displays the Application Security submenu.

Figure 6–6 Application Security Submenu

```
------------------------------------------------------
               Application Security

   Application name:  Inside COM, Chapter 11 Example
   Current Access permissions:  Custom
   Current Launch permissions:  Custom
   Current Configuration permissions:  Default

   1 - Use Default Access permission
   2 - Edit Custom Access permission
   3 - Use Default Launch permission
   4 - Edit Custom Launch permission
   5 - Use Default Configuration permission
   6 - Edit Custom Configuration permission

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Use Default Access permission`

  Sets the system to the default access permission values.

- `2 - Edit Custom Access permission`

  Displays the Registry Value Permissions submenu. This submenu allows you to view, add, modify, and delete access permission values for this application. For this set of submenus, see Section 6.3.2.

  The ACL Editor starts with the systemwide default values unless you previously set other values.

- `3 - Use Default Launch permission`

  Use the systemwide default launch permission values.

- `4 - Edit Custom Launch permission`

  Displays the Registry Value Permissions submenu. This submenu allows you to view, add, modify, and delete launch permission values for this application. For this set of submenus, see Section 6.3.2.

  The ACL Editor starts with the systemwide default values unless you previously set other values.

- `5 - Use Default Configuration permission`

  Use the systemwide configuration permission values.

- `6 - Edit Custom Configuration permission`

  The system displays the Registry Key Permissions submenu. This submenu allows you to view, add, modify, delete, and configure special access security permissions for this application. For this set of submenus, see Section 6.3.3.

6.3.2 Registry Value Permissions Submenus

To display this submenu:

1. From the DCOM$CNFG menu, choose option 1.
2. From the Applications List submenu, choose any application.
3. From the Application Properties submenu, choose option 2.
4. From the Application Security submenu, choose option 2 or 4.

Figure 6–7 Registry Value Permissions Submenu

```
------------------------------------------------------
               Registry Value Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Value: LaunchPermission
   Owner: Administrator

   Index    Name                          Type of Access
     1      OPENVMS_DCOM\USER1               Deny
     2      BUILTIN\Administrators           Allow
     3      Everyone                         Allow
     4      NT AUTHORITY\SYSTEM              Allow
     5      OPENVMS_DCOM\USER2               Allow

        (Index Number to Delete or Modify Access)
        (A to Add to list)

        (E to Exit to previous menu)
        (H for Help)

        Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `Index Number...`

  To change or delete an access type, enter the corresponding index number. The system displays Edit Registry Value Permissions submenu. See Figure 6–8.

- `A to Add to List`

  This option displays the Add Registry Value Permissions submenu. This submenu allows you to add a new entry to the OpenVMS Registry value's Access Control List. See Figure 6–9.

Figure 6–8 Edit Registry Value Permissions Submenu

```
------------------------------------------------------
               Edit Registry Value Permissions

       Application name: Inside COM, Chapter 11 Example
       Registry Value: AccessPermission
       Owner: Administrator

       Name: OPENVMS_DCOM\USER1
       Type of Access: Deny

       1 - Delete entry from list
       2 - Change Access

       (E to Exit to previous menu)
       (H for Help)

       Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Delete entry from list`

  Delete the entry from the Access Control List. If you delete all entries, you will deny access and launch permissions to everyone for the selected value.

- `2 - Change Access`

  Toggle the access type from Allow to Deny or Deny to Allow.

Figure 6–9 Add Registry Value Permissions Submenu

```
------------------------------------------------------
             Add Registry Value Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Value: LaunchPermission
   Owner: ROLLO

   1 - Add Specific User or Group
   2 - Add Everyone
   3 - Add NT AUTHORITY\System
   4 - Add BUILTIN\Administrators

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

— `1 - Add Specific User or Group`

Prompts for a user/group name and type of access. Specify the user name as *domain**username* or *username* if the account exists on the current domain.

— `2 - Add Everyone`

Allow or Deny `Everyone` Access/Launch permission to the application.

— `3 - Add NT AUTHORITY\System`

Allow or Deny `System` Access/Launch permission to the application.

— `4 - Add BUILTIN\Administrators`

Allow or Deny Administrator Access/Launch permission to the application.

When a user is part of two or more groups, Deny access takes precedence over Allow access.

6.3.3 Registry Key Permissions Submenus

To display this submenu:

1. From the DCOM$CNFG menu, choose option 1.
2. From the Applications List submenu, choose any application.
3. From the Application Properties submenu, choose option 2.
4. From the Application Security submenu, choose option 6.

Figure 6–10 Registry Key Permissions Submenu

```
------------------------------------------------------
               Registry Key Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Key: Inside COM, Chapter 11 Example
   Owner: Administrator

   Index    Name                    Type of Access

   1     BUILTIN\Administrators      Full Control
   2     NT AUTHORITY\SYSTEM         Full Control
   3     CREATOR OWNER               Full Control
   4     Everyone                    Special Access
   5     OPENVMS_DCOM\USER1          Read

   (Index Number to Delete or Modify Access)
   (A to Add to list)

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- Index Number...

  To change or delete an access type, enter the corresponding index number. The system displays Edit Registry Key Permissions submenu. See Figure 6–11.

- A to Add to List

  This option displays the Add Registry Key Permissions submenu. This submenu allows you to add a new entry to the OpenVMS Registry key's Access Control List. See Figure 6–13.

Figure 6–11 Edit Registry Key Permissions Submenu

```
------------------------------------------------------
               Edit Registry Key Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Key: Inside COM, Chapter 11 Example
   Owner: Administrator

   Name: BUILTIN\Administrators
   Type of Access: Full Control

   1 - Delete entry from list
   2 - Allow Full Control
   3 - Allow Read Access
   4 - Set/View Special Access

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Delete entry from list`

  Delete the entry from the security permissions list. If you delete all entries, noone can access the key and only the owner can change the permissions.

- `2 - Allow Full Control`

  Allow the user to access, to edit, and to take ownership of the key.

- `3 - Allow Read Access`

  Allow the user to read the key but not to save any changes to it.

- `4 - Set/View Special Access`

  Displays the Special Access Registry Key Permissions submenu. This submenu allows you to set customized permissions for the selected user or groups. See Figure 6–12.

Figure 6–12 Special Access Registry Key Permissions Submenu

```
------------------------------------------------------
               Special Access Registry Key Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Key: Inside COM, Chapter 11 Example

   Name: Everyone

   Type of Access                          Current Value

   0 - Query Value                           Yes
   1 - Set Value                             Yes
   2 - Create Subkey                         Yes
   3 - Enumerate Subkeys                     Yes
   4 - Notify                                Yes
   5 - Create Link                           No
   6 - Delete                                Yes
   7 - Write DACL                            No
   8 - Write Owner                           No
   9 - Read Control                          Yes

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `0 - Query Value`

  Allow the user to read a value from the key.

- `1 - Set Value`

  Allow the user to set one or more values for the key.

- `2 - Create Subkey`

  Allow the user to create subkeys on the key.

- `3 - Enumerate Subkeys`

  Allow the user to identify the subkeys of the key.

- `4 - Notify`

  Allow the user to audit notification events from the key.

- `5 - Create Link`

  Allow the user to create a symbolic link in the key.

- `6 - Delete`

  Allow the user to delete the key.

- `7 - Write DACL`

  Allow the user access to the key to write a discretionary ACL to the key.

- `8 - Write Owner`

  Allow the user access to the key to take ownership of the key.

- `9 - Read Control`

Allow the user access to the security information on the key.

Figure 6–13 Add Registry Key Permissions Submenu

```
------------------------------------------------------
               Add Registry Key Permissions

   Application name: Inside COM, Chapter 11 Example
   Registry Key: Inside COM, Chapter 11 Example
   Owner: Administrator

   1 - Add Specific User or Group
   2 - Add Everyone
   3 - Add NT AUTHORITY\System
   4 - Add BUILTIN\Administrators

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Add Specific User or Group`

  Prompts for a user/group name and type of access. Specify the user name as *domain**username* or *username* if the account exists on the current domain.

- `2 - Add Everyone`

  Allow `Everyone` Full Control or Read Access to the application.

- `3 - Add NT AUTHORITY\System`

  Allow `System` Full Control or Read Access to the application.

- `4 - Add BUILTIN\Administrators`

  Allow `Administrator` Full Control or Read Access to the application.

6.3.4 Application Identity Submenu

To display this submenu:

1. From the DCOM$CNFG menu, choose option 1.
2. From the Applications List submenu, choose any application.
3. From the Application Properties submenu, choose option 3.

The system displays the Application Identity submenu.

Figure 6–14 Application Identity Submenu

```
------------------------------------------------------
               Application Identity

   Which user account do you want to use to run this application?

   Application name:  Inside COM, Chapter 11 Example
   Current Identity:  NTLM Account OPENVMS_DCOM\USER2

   1 - Launching User
   2 - NTLM Account
   3 - OpenVMS Username
   4 - OpenVMS DCOM Guest Account

   (E to Exit to previous menu)
   (H for Help)

   Please enter account you wish to use:
------------------------------------------------------
```

The options are as follows:

- `1 - Launching User`

  Specifies that the application will run using the security context of the user who started the application. This is the default if NTLM security is available.

- `2 - NTLM Account`

  Specifies that the application will run using the security context of the specified NTLM account. If you specify a valid User/Group name, the system prompts you for a password. The system checks that the password matches the password you used to log on (through `NTA$LOGON`). If the passwords do not match, you can either continue and write this new password to the OpenVMS Registry or reenter a password that matches your logon password.

  Note

  If you enter a new password, the system does not synchronize the new password with any other password. You must synchronize the passwords manually.

  You must have the IMPERSONATE privilege for the password to be validated.

  You must have system write access (SYSPRV or REG$UPDATE) to the OpenVMS Registry to write the password to the database.

- `3 - OpenVMS Username`

  Specifies that the application will run using the security context of the specified OpenVMS account. This option is active only when you are using unauthenticated COM for OpenVMS.

- `4 - OpenVMS DCOM Guest Account`

  Specifies that the application will run using the security context of the OpenVMS DCOM Guest account. This option is active only when you are using unauthenticated COM for OpenVMS. If you are using unauthenticated COM for OpenVMS, this option is the default.

6.3.5 The DCOM$CNFG System-wide Default Properties Submenu

To display this submenu, from the DCOM$CNFG Main menu, choose option 2.

The system displays the System-wide Default Properties submenu.

Figure 6–15 System-wide Default Properties Submenu

```
------------------------------------------------------
            System-wide Default Properties

   1 - Enable Distributed COM on this computer (Yes/No)
             Current value: Yes
   2 - Default Authentication Level
   3 - Default Impersonation Level

   (E to Exit to previous menu)
   (H for Help)

   Please enter your choice:
------------------------------------------------------
```

The options are as follows:

- `1 - Enable Distributed COM on this computer (Yes/No)`

  Enables or disables COM on this computer.

- `2 - Default Authentication Level`

  Sets packet-level security on communications between applications. This systemwide default applies to all applications installed on this computer.

Figure 6–16 Default Authentication Level Submenu

```
-------------------------------------------------------------------
                Default Authentication Level

   The Authentication Level specifies security at the packet level.

   Current value: Connect

      1 - Default
      2 - None
      3 - Connect
      4 - Call
      5 - Packet
      6 - Packet Integrity

      (E to Exit to previous menu)
      (H for Help)

    Please enter your choice:
-------------------------------------------------------------------
```

  Enter a number to select the desired Authentication level. When installed, the system default for the Default Authentication Level is `Connect`.

- `3 - Default Impersonation Level`

  Specifies whether applications can determine who is calling them, and whether the application can perform operations using the client's identity.

Figure 6–17 Default Impersonation Level Submenu

```
------------------------------------------------------------------
               Default Impersonation Level

   The Impersonation Level specifies whether applications can determine
   who is calling them, and whether the application can perform
   operations using the client's identity.

   Current value: Identify

      1 - Anonymous
      2 - Identify
      3 - Impersonate

      (E to Exit to previous menu)
      (H for Help)

   Please enter your choice:
------------------------------------------------------------------
```

Enter a number to select the desired Impersonation level. When installed, the system default for the Default Impersonation Level is `Identify`.

6.3.6 System-wide Default Security Submenu

To display this submenu, from the DCOM$CNFG Main Menu, choose option 3.

The system displays the System-wide Default Security submenu.

Figure 6–18 System-wide Default Security Submenu

```
------------------------------------------------------------------
               System-wide Default Security

        1 - Access Permissions Default
        2 - Launch Permissions Default
        3 - Configuration Permissions Default

        (E to Exit to previous menu)
        (H for Help)

        Please enter your choice:
------------------------------------------------------------------
```

The options are as follows:

- `1 - Access Permissions Default`:

  Displays the Registry Value Permissions submenu. This submenu allow you to view, add, modify, and delete Access permission values for the systemwide default for all applications.

- `2 - Launch Permissions Default`:

  Displays the Registry Value Permissions submenu. This submenu allows you to view, add, modify, and delete Launch Permission Values for the systemwide default for all applications. You must restart the COM for OpenVMS Service Control Manager for the new setting to take effect.

- `3 - Configuration Permissions Default`:

  Displays the security permission values for the HKEY_CLASSES_ROOT Registry key.

When you first install the system, by default only Administrator and System accounts have application launch and access permissions. HP recommends that you do not change these default settings. Typically you modify an individual application's launch and access security to grant or deny permissions to Everyone, various Groups, or even specific users. HP recommends this technique over adjusting the machinewide default security settings that affect all applications.

6.4 Registering In-Process Servers: DCOM$REGSVR32 Utility

All COM components (implemented as either an out-of-process server or as an in-process server) must be registered in the OpenVMS Registry before you can use them.

Out-of-process servers, which are implemented as executable programs (.EXE files), usually contain code to register and unregister the components contained within them. The advantage an out-of-process server has over an in-process server is that you can run the executable and automatically create the necessary registry keys.

In-process servers, which are usually implemented as dynamic link libraries (.DLL files) on Windows or as shareable images on OpenVMS, also contain code to register and unregister the components within them automatically. However, these in-process servers cannot be run the same way as an executable image because they do not contain a main entry point. As a result, you must manually register the components contained within a .DLL, or create a command procedure to perform the registration.

Microsoft provides the REGSVR32 utility that you can use to register the components contained within a DLL. REGSVR32 takes as a command line argument the following:

- DLL name
- Switches to register or unregister the components

When registering a DLL's components, REGSVR32 searches the specified DLL for the DllRegisterServer symbol and, if found, calls it. When unregistering a DLL, REGSVR32 calls DllUnregisterServer. This means that all in-process components that you want to register automatically must include these two entry points in their export files.

To facilitate the registration of components contained within shareable images on OpenVMS systems, HP created the DCOM$REGSVR32 utility. The DCOM$REGSVR32 utility does the same things that the Microsoft REGSVR32 utility does. Any shareable images that contain components to be registered must also include the DllRegisterServer and DllUnregisterServer universal symbols in their symbol vectors. Both the DCOM$REGSVR32 and the REGSVR32 utilities use the same command line syntax.

During the COM for OpenVMS installation, the system places the DCOM$REGSVR32.EXE file in the SYS$SYSTEM directory.

Before you use the DCOM$REGSVR32 utility, you must define a symbol that allows the utility to accept foreign command lines. For example:

```
$ regsvr32 :== $DCOM$REGSVR32
```

You can use either method to activate the utility, and register or unregister components contained in shareable images.

To display help for DCOM$REGSVR32, enter the following:

```
$ regsvr32 -?
```

Table 6–1 summarizes the DCOM$REGSVR32 command line options.

Table 6–1 DCOM$REGSVR32 Command Line Options

Switch	Use
-?, /?	Display help file (this table).
shareable-image-name	Register the specified shareable image name.
-u or /u *image-name*	Unregister the specified shareable image name.

Note

The DCOM$REGSVR32 utility requires that the shareable image name contain a full directory specification.

Example 6–4 demonstrates how to register an in-process component (contained within a shareable image) using the DCOM$REGSVR32 utility.

Example 6–4 Registering a Component Using the DCOM$REGSVR32 Utility

```
$ regsvr32 USER$DISK:[SEYMOUR.DISPATCH_SAMPLE1]CMPNT$SHR.EXE
Class factory:          Create self.
DllRegisterServer:      Registering Server DLL
Creating key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}
Creating key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}\InProcServer32
Creating key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}\ProgID
Creating key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}\VersionIndependentProgID
Creating key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}\TypeLib
Creating key InsideCOM.Chap11
Creating key InsideCOM.Chap11\CLSID
Creating key InsideCOM.Chap11\CurVer

Creating key InsideCOM.Chap11.1
Creating key InsideCOM.Chap11.1\CLSID
Class factory:          Destroy self.
```

Example 6–5 demonstrates how to unregister an in-process component (contained within a shareable image) using the DCOM$REGSVR32 utility.

Example 6–5 Unregistering a Component Using the DCOM$REGSVR32 Utility

```
$ regsvr32 /u USER$DISK:[SEYMOUR.DISPATCH_SAMPLE1]CMPNT$SHR.EXE

Class factory:          Create self.
DllUnregisterServer:    Unregistering Server DLL
Deleting key InProcServer32
Deleting key ProgID
Deleting key VersionIndependentProgID
Deleting key TypeLib
Deleting key LocalServer32
Deleting key CLSID\{0C092C2C-882C-11CF-A6BB-0080C7B2D682}
Deleting key CLSID
Deleting key CurVer
Deleting key InsideCOM.Chap11
Deleting key CLSID
Deleting key InsideCOM.Chap11.1
Class factory:          Destroy self.
```

7

Developing a COM for OpenVMS Application

This chapter explains how to develop COM applications for OpenVMS.

Note

You can find the sample COM applications shown in this chapter in the following directories on the COM for OpenVMS kit:

```
DCOM$EXAMPLES:[SAMPLE1]
DCOM$EXAMPLES:[SIMPLE]
DCOM$EXAMPLES:[DISPATCH_SAMPLE1]
```

SAMPLE1 and DISPATCH_SAMPLE1 are taken from Dale Rogerson's book, *Inside COM*, published by Microsoft Press. This book is a good reference for developing COM applications.

The following sections describe how to create a COM for OpenVMS application.

Note

Building COM for OpenVMS applications places demands on the virtual memory requirements of a process. You should have a minimum page file quota of 100,000 pagelets before building a COM for OpenVMS application. This is a HP C++ compiler requirement.

7.1 Step 1: Generate Unique Identifiers

Use the DCOM$GUIDGEN utility to generate 16-byte globally unique identifiers (GUIDs).

For example:

```
$ SET COMMAND DCOM$LIBRARY:DCOM$GUIDGEN.CLD
$ DCOM$GUIDGEN [/FORMAT=value] -
_$ [/COUNT=value] [/OUTPUT=value]
```

The following table summarizes the GUID format options.

OpenVMS qualifier (*value*)	UNIX switch	Use
IDL	-i	Output GUID in an IDL interface template.
STRUCT	-s	Output GUID as an initialized C struct.
IMPLEMENT_ OLECREATE	-c	Output GUID in IMPLEMENT_OLECREATE(...) format.

OpenVMS qualifier (*value*)	UNIX switch	Use
DEFINE_GUID	-d	Output GUID in DEFINE_GUID(...) format.
GUID_STRUCT	-g	Output GUID as an initialized static const GUID struct.
REGISTRY_GUID	-r	Output GUID in registry format.

Note

The last four options in the preceding table are the same as the four options in the Windows Guidgen utility.

The following table lists additional options supported by the DCOM$GUIDGEN utility.

OpenVMS qualifier	UNIX switch	Use
/OUTPUT=*filename*	-o *filename*	Redirect output to a specified file.
/COUNT=*number*	-n *number*	Number of GUIDs to generate.
not available	-h, -?	Display command option summary.

You can specify more than one format for the same GUID.

7.2 Step 2: Build an Application Using the MIDL Compiler

The following sections describe how to use the MIDL compiler to build an application.

7.2.1 Running the MIDL Compiler

The MIDL compiler consists of the following separate images:

- SYS$SYSTEM:DCOM$MIDL.EXE

  The executable image that takes its arguments (parameters) from the DCL command line.

- SYS$SHARE:DCOM$MIDL_SHR.EXE

  A shareable image library that does the actual work for DCOM$MIDL.EXE.

To run MIDL, you must first define a DCL symbol. For example:

```
$ midl :== $dcom$midl
$ midl -?
$ midl -Oicf -idcom$library: example.idl
```

The midl -? command displays a list of valid command-line arguments. For a list of these arguments, see Appendix A.

7.2.2 Running the MIDL Compiler with DCOM$RUNSHRLIB

The DCOM$MIDL.EXE utility gets its arguments from the DCL foreign command-line buffer. DCL foreign commands can have a maximum of 255 characters.

Because of the number of arguments that DCOM$MIDL.EXE can accept, you might exceed this maximum number of characters if you specify a complex MIDL command (for example, a command that contains mixed-case arguments that require quotation marks).

As a workaround, you can use the SYS$SYSTEM:DCOM$RUNSHRLIB.EXE utility. Use the following procedure:

1. Define the DCL command DCOM$RUNSHRLIB.

   A process that needs to use DCOM$RUNSHRLIB.EXE must first use the OpenVMS DCL Command Definition utility to define the DCL command DCOM$RUNSHRLIB. For example:

   ```
   $ SET COMMAND DCOM$LIBRARY:DCOM$RUNSHRLIB.CLD
   ```

   DCOM$LIBRARY:DCOM$RUNSHRLIB.CLD defines the DCOM$RUNSHRLIB DCL command. The following table shows the command's parameters.

Argument	Value	Required/Optional
P1	Name of the shareable image library. This can be a logical name, the name of an image in SYS$SHARE:, or a full file specification.	Required
P2	Name of the routine to be called as a C or C++ main( ) routine with an argc/argv vector.	Required
P3	List of qualifiers, in quotation marks.	Optional

2. Define the DCL symbol midl to use DCOM$RUNSHRLIB.EXE to parse the command line and call the DCOM$MIDL_MAIN function in the DCOM$MIDL_SHR shareable image library. For example:

   ```
   $ midl :== DCOM$RUNSHRLIB DCOM$MIDL_SHR DCOM$MIDL_MAIN
   ```

   The new DCL command MIDL accepts multiple command-line arguments inside a single quoted string. If the command becomes too long, you can specify multiple quoted strings, using a comma to separate the strings.

   For example, here is a complex MIDL command that fails:

   ```
   $ midl :== $dcom$midl
   $ midl -Zp8 -Oicf -Os -oldnames -char unsigned -
      -error allocation -error bounds_check -error stub_data -
      -ms_ext -c_ext -out [.OBJ] -
      -I[INC] -I[PROJECT_WIDE_INC] -I[COMMON_INC] -IDCOM$LIBRARY: -
      -DRMS_DB "-DOpenVMS_Definitions" "-DPermanentProcess" -
      -header [.obj]example.h -client none -server none example.idl
   %DCL-W-TKNOVF, command element is too long - shorten
   ```

   You can successfully specify this command using DCOM$RUNSHRLIB as follows:

   ```
   $ set command dcom$library:dcom$runshrlib.cld
   $ midl :== DCOM$RUNSHRLIB DCOM$MIDL_SHR DCOM$MIDL_MAIN
   $ midl "-Zp8 -Oicf -Os -oldnames -char unsigned",-
      "-error allocation -error bounds_check -error stub_data",-
      "-ms_ext -c_ext -out [.OBJ]",-
      "-I[INC] -I[PROJECT_WIDE_INC] -I[COMMON_INC] -IDCOM$LIBRARY:",-
      "-DRMS_DB -DOpenVMS_Definitions -DPermanentProcess",-
      "-header [.obj]example.h -client none -server none example.idl"
   ```

7.2.3 Modifying Your Applications To Use the C++ Only MIDL Compiler

By default, the MIDL compiler included in COM for OpenVMS generates IDL files with a .CXX extension. If you previously specified names for the generated files in your MIDL command and want to use the COM for OpenVMS MIDL compiler changes, you need to update your command line.

For example, if you previously generated IDL files with the following command:

```
MIDL -Oicf server.idl -IDCOM$LIBRARY: -dlldata DLLDATA.C
```

Change your command to the following:

```
MIDL -Oicf server.idl -IDCOM$LIBRARY: -dlldata DLLDATA.CXX
```

If you want to continue to use the C compiler to build your generated files, simply specify them as .C files on your command line.

Note

If you previously accepted the default file names, and want to continue to use the C compiler to build the IDL files, you must specify every IDL file on the command line.

For example, if your MIDL command was similar to the following:

```
MIDL -Oicf server.idl -IDCOM$LIBRARY: -dlldata DLLDATA.C
```

Change it to the following:

```
MIDL -Oicf server.idl -IDCOM$LIBRARY: -dlldata DLLDATA.C -guids
     GUIDS.C -proxy SERVER_P.C
```

7.2.4 Required MIDL Switches

When running MIDL on OpenVMS, you must specify the `-Oicf` MIDL command-line switch.

7.2.5 Required Include Directories

MIDL components typically import `UNKNWN.IDL`, which contains the component definitions for `IUnknown` and `IClassFactory`. `UNKNWN.IDL` and other COM-related IDL and header files are located in `DCOM$LIBRARY`. To build your component's IDL file, use the following switch:

```
-IDCOM$LIBRARY:
```

7.3 Step 3: Compile the COM Application

The following sections describe how to compile COM for OpenVMS applications.

Note

COM application developers need access to the OpenVMS Registry to register and control access to a given application. For more information about OpenVMS Registry privileges, see Section 12.6.1 and Section 6.3.

7.3.1 Required Header File: VMS_DCOM.H

The VMS_DCOM.H header file contains macro definitions that enable your COM for OpenVMS application to compile properly. You must include this header file as the first uncommented line in every source file and header file you create.

The MIDL compiler for OpenVMS includes VMS_DCOM.H in all output files that it generates.

7.3.2 Required Macro Definitions

Be sure to always include the following /DEFINE qualifier in all of your CXX commands:

```
/DEFINE=(UNICODE,_WIN32_DCOM)
```

The UNICODE macro ensures that the wide character variants of Win32 APIs and data structures are enabled when you compile your code. (This macro is also defined in VMS_DCOM.H.)

The _WIN32_DCOM macro definition is recognized by the header files and is required to ensure the proper definition of structures and COM APIs.

7.3.3 Required Include Directories

COM for OpenVMS applications typically require header files that come from DCOM$LIBRARY.

Include the following qualifier in your C and CXX commands:

```
/INCLUDE=DCOM$LIBRARY
```

If you already have an /INCLUDE qualifier in your command line, modify the command to include DCOM$LIBRARY.

7.3.4 Required C++ Qualifiers

You must specify the following C++ qualifiers when you build COM for OpenVMS applications:

- /EXCEPTIONS=CLEANUP

  Specify the /EXCEPTIONS=CLEANUP qualifier on C++ commands to enable C++ exceptions.

- /STANDARD=CFRONT

  The C++ compiler supports many different compilation standards. HP recommends that you use /STANDARD=CFRONT.

  /STANDARD=CFRONT informs the compiler that it should follow the language conventions defined in the AT&T cfront implementation.

7.4 Step 4: Link the COM Application

To build a COM for OpenVMS application, you must build both client and component images. Because you can implement a component as either an in-process component or an out-of-process component, you must build either a shareable image or an executable image, or both. If you are creating a new interface, you must also build a proxy/stub shareable image, unless you are using the IDispatch interface. In that case, the Automation Marshaler will be used instead of the proxy/stub shareable image. The proxy/stub shareable image provides an interface-specific object that packages parameters for that interface in preparation for a remote method call. A proxy runs in the sender's address

space and communicates with a corresponding stub in the receiver's address space.

The following sections describe the steps you must follow to link the client, component, and proxy/stub images.

7.4.1 Linking the Client and the Out-of-Process Component

Although you do not need to specify any qualifiers to link the client or the component executable images, you must link both images with the DCOM OLE32 shareable image (to satisfy references to COM APIs).

The specific link-time dependency is DCOM$LIBRARY:DCOM.OPT.

If you have one or more C++ modules, use the C++ linker (CXXLINK) instead of the standard OpenVMS linker so you can specify the location of your C++ repository (/CXX_REPOSITORY qualifier). For example:

```
$  CXXLINK/your-specific-linker-qualifiers list-of-object-modules, -
_$ DCOM$LIBRARY:DCOM.OPT/OPTIONS -
_$ application.OPT/OPTIONS /REPOSITORY=[.CXX_REPOSITORY]
```

Other ways of including the options file are as follows:

- Include the list of object modules in an options file instead of on the command line.
- Use DCOM$LIBRARY:DCOM.OPT.

7.4.2 Linking the In-Process Component Shareable Image

The specific link-time dependency is DCOM$LIBRARY:DCOM.OPT.

7.4.2.1 Creating a Symbol Vector

Linking the in-process component shareable image requires that you create a symbol vector for the entry points that COM for OpenVMS expects to call within the shareable image. The Win32 run-time environment enforces a naming standard on the DllMain entry point, which must contain the following:

- Actual entry point name
- A suffix that includes the image name or a portion of the image name, depending on the format of the image name.

  If the image name ends in $SHR (for example, CMPNT$SHR), the suffix is the image name up to and including the dollar sign ($).

  If the image name ends in anything other than $SHR (for example, CMPNT_SHARE), the suffix is the full image name.

For example, a component shareable image with the name CMPNT$SHR would define the symbol vector using the following options file:

```
!
! The list of symbols exported by CMPNT$SHR.EXE.
!
SYMBOL_VECTOR=(-
        _DllMain_CMPNT$/DllMain    = PROCEDURE,-
        DllGetClassObject              = PROCEDURE,-
        DllCanUnloadNow                = PROCEDURE,-
        DllRegisterServer              = PROCEDURE,-
        DllUnregisterServer            = PROCEDURE)
```

A component shareable image with the name CMPNT_SHARE would define the symbol vector using the following options file:

```
!
! The list of symbols exported by CMPNT_SHARE.EXE.
!
SYMBOL_VECTOR=(-
        _DllMain_CMPNT_SHARE/DllMain  = PROCEDURE,-
        DllGetClassObject               = PROCEDURE,-
        DllCanUnloadNow                 = PROCEDURE,-
        DllRegisterServer               = PROCEDURE,-
        DllUnregisterServer             = PROCEDURE)
```

7.4.3 Linking the Proxy/Stub Shareable Image

The specific link-time dependency is SYS$LIBRARY:DCOM$RPCRT4_SHR.EXE.

7.4.3.1 Creating a Symbol Vector

Linking the proxy/stub shareable image is more involved because you must create a symbol vector for the entry points that COM for OpenVMS expects to call within the shareable image. The Win32 run-time environment enforces a naming standard on the DllMain entry point, which must contain the following:

- Actual entry point name
- A suffix that includes the image name or a portion of the image name, depending on the format of the image name.

  If the image name ends in $SHR (for example, PROXY$SHR), the suffix is the image name up to and including the dollar sign ($).

  If the image name ends in anything other than $SHR (for example, PROXY_SHARE), the suffix is the full image name.

For example, a proxy/stub shareable image with the name PROXY$SHR would define the symbol vector using the following options file:

```
!
! RPC Shareable Image
!
 SYS$LIBRARY:DCOM$RPCRT4_SHR.EXE/SHARE
!
!
! The list of symbols exported by PROXY$SHR.EXE.
!
SYMBOL_VECTOR=(-
        _DllMain_PROXY$/DllMain   = PROCEDURE,-
        DllGetClassObject               = PROCEDURE,-
        DllCanUnloadNow                 = PROCEDURE,-
        GetProxyDllInfo                 = PROCEDURE,-
        DllRegisterServer               = PROCEDURE,-
        DllUnregisterServer             = PROCEDURE)
```

A proxy/stub shareable image with the name PROXY_SHARE would define the symbol vector using the following options file:

```
!
! RPC Shareable Image
!
 SYS$LIBRARY:DCOM$RPCRT4_SHR.EXE/SHARE
!
!
! The list of symbols exported by PROXY_SHARE.EXE.
!
SYMBOL_VECTOR=(-
        _DllMain_PROXY_SHARE/DllMain = PROCEDURE,-
        DllGetClassObject                = PROCEDURE,-
        DllCanUnloadNow                  = PROCEDURE,-
        GetProxyDllInfo                  = PROCEDURE,-
        DllRegisterServer                = PROCEDURE,-
        DllUnregisterServer              = PROCEDURE)
```

7.5 Required OpenVMS Registry Entries

The following sections list and describe the required OpenVMS Registry entries.

7.5.1 HKEY_CLASSES_ROOT\CLSID

The CLSID subkey contains all CLSIDs for the components supported on your system. You must register your components' CLSIDs here. Each registered CLSID should contain the following:

- An unnamed value whose type is a zero-terminated string with a data value describing the component.
- A named value, `AppID`, whose type is a zero-terminated string with a data value that is the CLSID of the component.

7.5.1.1 Component CLSIDs

A class identifier (CLSID) is a globally unique identifier (GUID) associated with an OLE class object. COM for OpenVMS server applications typically register their CLSIDs in the OpenVMS Registry so clients can locate and load the executable code associated with the OLE class object.

Register the CLSID for the component under the subkey `HKEY_CLASSES_ROOT\CLSID`.

A component CLSID registration should contain the following subkeys:

- `LocalServer32`

  This key's value should contain a zero-terminated string with a data value that is the location of the out-of-process server executable.

- `ProgID`

  This key's value should contain a zero-terminated string with a data value that is the programmatic ID of the CLSID. These are typically in the format *program.component.version*.

- `VersionIndependentProgID`

  This key's value should contain a zero-terminated string with a data value that is the programmatic ID (less the version number) of the CLSID. These are typically in the format *program.component*.

- `InProcServer32`

  This key's value should contain a zero-terminated string with a data value that is the location of the in-process server's shareable image.

- Type Libraries

  Type libraries are important for implementing the IDispatch interface. A type library registers itself when it calls the OLE Automation RegisterTypeLib run-time routine. You must also add a Typelib subkey under your component's CLSID. The Typelib subkey contains your type library's GUID. For example, the following key should contain your LIBID:

  HKEY_CLASSES_ROOT\CLSID*{GUID}*\TYPELIB {value=LIBID}

7.5.1.2 Proxy/Stub CLSIDs

The proxy/stub shareable image provides an interface-specific object for packaging parameters for that interface. Because the proxy/stub shareable image contains an object, it needs a CLSID and it needs to be included in the OpenVMS Registry. You must register a CLSID for the proxy in the OpenVMS Registry the same way as the CLSID for the component.

The CLSID for the proxy should be registered under the subkey HKEY_CLASSES_ROOT\CLSID.

A proxy/stub CLSID registration should contain the following subkey:

- InProcServer32

  The InProcServer32 value should contain a zero-terminated string with a data value that is the location of the proxy/stub shareable image. The proxy/stub CLSID and its subkey enable COM to locate the proxy/stub shareable image.

7.5.2 HKEY_CLASSES_ROOT\Interface

The Interface subkey contains all interfaces registered with the system. You must register the component's interface IDs (IIDs) in this subkey.

Each interface registered contains at least one of the following subkeys:

- NumMethods

  The NumMethods value should contain a zero-terminated string with a data value that is the number of methods contained in the interface.

- ProxyStubClsid32

  The ProxyStubClsid32 value should contain a zero-terminated string with a data value that is the CLSID of the proxy/stub shareable image. This CLSID should be the same as that described in Section 7.5.1.2.

7.6 Converting OpenVMS and Windows Error Codes to Text

As you develop and test COM components, you will find that the OpenVMS and Windows systems return seemingly indecipherable error codes. To help you make these codes more understandable, HP has included some ways to translate them.

7.6.1 NTA$VMSGetMessage

HP has included the NTA$VMSGetMessage routine to translate error codes into displayable text. The following section describes the NTA$VMSGetMessage routine.

To implement this routine, you must include the NTA_MESSAGE.H file in the DCOM$LIBRARY: directory and link with the DCOM$LIBRARY:NTA_GETMSG.OBJ object module.

The NTA$VMSGetMessage routine, described in the next section, translates error codes into displayable text. The input error code must be one of the following:

- An OpenVMS error code
- A Windows HRESULT
- A Windows Win32 error code
- A Windows status code set as “user defined”

NTA$VMSGetMessage

The NTA$VMSGetMessage routine translates error codes into displayable text.

Format

Return=NTA$VMSGetMessage (status, text, flag, [count])

Arguments

status

OpenVMS usage: error_code
type: longword (unsigned)
access: read only
mechanism: by value

This status field must be one of the following:

Input Error Code	Example
OpenVMS error code	0x074AA6BA
Windows HRESULT	0x80070031
Windows Win32 error code	0x00000031
Windows status code with the user-defined bit set	0xE74AA6BA

If the security API returns a Windows status code, the format of the status field is an OpenVMS status code OR'd with the Windows status control bits set. For example:

Input Error Code	Result
OpenVMS error code	0x074AA6BA
Windows status code	0xE74AA6BA

text

OpenVMS usage: error_text
type: character string
access: write
mechanism: by reference

This argument is a NULL terminated string that contains the returned text from the SYS$GETMSG system service. The maximum size returned (as defined by the SYS$GETMSG system service) is 256 bytes. To avoid overwriting memory, the caller must provide a buffer address of at least 257 bytes.

flag

OpenVMS usage: flag
type: longword (unsigned)
access: read only
mechanism: by value

Controls the translation of the error code. The following values are defined in NTA_MESSAGE.H:

NTAWIN$_UNKNOWN	Unknown error code
NTAWIN$_VMS	OpenVMS error code
NTAWIN$_NT	Windows HRESULT error code
NTAWIN$_WINDOWS	Windows Win32 error code
NTAWIN$_USER	Windows status code

If you provide the value NTAWIN$_UNKNOWN, the routine makes its best estimate as to the correct text. The routine parses the text as follows:

1. Check for a Windows HRESULT (high-order nibble = 0x8). If this check fails, go to the next step.
2. Check for a Windows user-defined status code (high-order nibble = 0xE). If this check fails, go to the next step.
3. Assume this is an OpenVMS error code.

   The system cannot tell the difference between an OpenVMS error code and a Windows Win32 error code.

count

OpenVMS usage:	FAO count
type:	longword (unsigned)
access:	write
mechanism:	by reference

This argument is the optionally returned FAO argument count in the returned message. Currently all NTAWIN messages use ASCII substitution arguments (!AS) only. The caller must convert all numeric data to ASCII before performing the substitution with SYS$FAO.

Description

This routine uses the OpenVMS SYS$GETMSG system service. The messages are stored in the SYS$MESSAGE:NTAWINMSG.EXE and SYS$MESSAGE:NTARPCMSG.EXE images.

To call this routine, you must include the NTA_MESSAGE.H file in the DCOM$LIBRARY: directory and link with the SYS$LIBRARY:DCOM$WIN32_SHR shareable image.

Condition Values Returned

Any status from the SYS$GETMSG system service.

For more information about the SYS$GETMSG system service, see the *OpenVMS System Services Reference Manual*.

7.6.2 DCOM$TOOL SHOW ERROR

HP has included command-line syntax to convert error codes into displayable text. The following section describes the DCOM$TOOL SHOW ERROR syntax.

To use the DCOM$TOOL utility to convert the codes, use any of the following methods:

- Enter the following command to run the DCOM$TOOL utility:

```
$ RUN SYS$SYSTEM:DCOM$TOOL.EXE
```

- Define a DCL symbol to use DCOM$TOOL and specify parameters on the command line. For example:

```
$ DCOMTOOL :== $SYS$SYSTEM:DCOM$TOOL.EXE
$ DCOMTOOL
```

You can specify parameters for any of these methods on the command line. Table 7–1 shows the DCOM$TOOL utility command line parameters. If you do not specify any parameters, the system prompts you for the required information.

Table 7–1 DCOM$TOOL Utility Command Line Parameters

Argument	Value	Required or Optional
P1	Command verb : SHOW	Required
P2	Command adjective : ERROR	Required
P3	Error code in DCL number format (%X)	Required
P4	Optional qualifers	Optional

The following example shows a typical DCOM$TOOL session to translate error codes:

```
$ DCOMTOOL :== $DCOM$TOOL.EXE
$ DCOMTOOL SHOW ERROR %x80070005
```

7.6.2.1 DCOM$TOOL Optional Qualifiers

DCOM$TOOL accepts the following optional qualifiers:

- /VMS_ERROR

  This is a OpenVMS error code. An example of an OpenVMS error code is as follows:

```
%x074AA6BA
```

- /HRESULT

  This is a Windows HRESULT. An example of a Windows HRESULT is as follows:

```
%x80070031
```

- /WIN32

  This is a Win32 error code. An example of a Win32 error code is as follows:

```
%x00000031
```

- /USER_DEFINED

This is a Windows status code with the user-defined bit set. An example of a User Defined error code is as follows:

```
%xE74AA6BA
```

Note

The DCOM$TOOL utility SHOW ERROR feature follows the rules, restrictions, and guidelines of the OpenVMS Message Utility. For more information, see the *OpenVMS Command Definition, Librarian, and Message Utilities Manual*.

8
Authentication

8.1 Authentication Overview

Authentication is the act of verifying a user's identity by the computer system before permitting access to the system. After successfully authenticating a user, the system binds the user's authorization information to the user's process in the form of *credentials*. The system uses these credentials to determine whether to grant or deny access to system resources.

OpenVMS provides both native (SYSUAF-based) and Windows compatible authentication and authorization capabilities as follows:

- Native: The system performs authentication using password information stored in the SYSUAF.DAT file. Authorization information consists of UIC, privileges, and rights identifiers.
- Windows: The system performs authentication using password information stored in a SAM database managed by domain controllers. Authorization information consists of primary SID, group SIDs, session key, and privileges obtained from the user's account information in the SAM database.

After OpenVMS successfully authenticates a user (either native or Windows), OpenVMS attaches the user's native credentials to the process using a structure known as a *persona*. If the system used Windows for authentication, OpenVMS also attaches the user's Windows credentials to the process (as an extension to the persona).

8.2 Acquiring Windows Credentials Using NTA$LOGON

NTA$LOGON is a utility that allows you to acquire NTLM credentials. All processes that need Windows security to access the OpenVMS Registry or COM for OpenVMS facilities require NTLM credentials.

You must provide NTA$LOGON with a user account name, a password, and (if required) a domain name. NTA$LOGON uses the Authentication and Credential Management (ACM) Authority to contact the domain controller and acquire a Windows access token. NTA$LOGON merges the Windows information with the user's OpenVMS credentials.

For a detailed review of NTA$LOGON dependencies and a description of how NTA$LOGON interacts with other parts of the OpenVMS infrastructure, see Section 5.1 and Section 4.10 (especially the ACME server and HP Advanced Server for OpenVMS server).

To use the NTA$LOGON utility, you can enter any of the following:

- Enter the following command to run the NTA$LOGON utility:

```
$ RUN SYS$SYSTEM:NTA$LOGON
```

  The system prompts you for a user account name and password.

- Define a DCL symbol to use NTA$LOGON to parse the command line. For example:

```
$ NTLOGON :== $NTA$LOGON
$ NTLOGON
```

You can specify parameters on the command line. Table 8–1 shows the NTA$LOGON utility command-line parameters. If you do not specify any parameters, the system prompts you for the required information.

Table 8–1 NTA$LOGON Utility Command Line Parameters

Argument	Value	Required/Optional
P1	User account name. If an account name is needed but was not specified on the command line, NTA$LOGON prompts for input.	Optional
P2	Password. If a password is needed but was not supplied on the command line, NTA$LOGON prompts for input (echoing suppressed).	Optional

Example 8–1 shows a typical NTA$LOGON session to acquire credentials.

Example 8–1 Sample NTA$LOGON Session

```
$ NTLOGON :== $NTA$LOGON
$ NTLOGON joesmith
Password:
```

Note

Windows domain names and user account names are not case sensitive. NTA$LOGON converts all domain names and user account names to uppercase. If you specify a password on the command line, DCL converts all characters to uppercase, unless you enclose the password in quotation marks ("").

8.2.1 NTA$LOGON Optional Qualifiers

NTA$LOGON accepts the following optional qualifiers:

- **/DELETE**

  Deletes the current Windows credentials.

  If you specify the **/DELETE** qualifier with the **/WRITE_FILE** qualifier, the system deletes the password record for the specified domain name and user account name from the file.

- **/DOMAIN=***name*

  Specifies a domain name. This qualifier converts the name to uppercase. If you do not specify this qualifier, the system uses the default domain name.

- **/LIST**

  Lists the domain name and the user account name assigned to the current process.

If you use the **/LIST** qualifier with the **/READ_FILE** or **/WRITE_FILE** qualifier, the system lists the contents of the file.

- **/LOG**

  Displays a message when an operation completes.

- **/OVERRIDE_MAPPING**

  Acquires Windows credentials for the specified Windows user account name even if the OpenVMS user name of the process does not match the OpenVMS user name associated with that Windows user account name in the domain controller.

  This qualifier requires the IMPERSONATE privilege.

- **/READ_FILE** [=*file*]

  This qualifier causes the system to search the binary input file created by the **/WRITE_FILE** qualifier for the specified domain name and user account name, instead of reading the password from the user input device. The **/READ_FILE** qualifier supports only binary files created by the NTA$LOGON/WRITE_FILE command.

  If the system finds a matching record, NTA$LOGON attempts to use that password to acquire Windows credentials.

  If you do not provide a file specification, the system uses the following default file specification:

  ```
  DCE$COMMON:[000000]NTA$LOGON.DAT
  ```

- **/TYPE={BATCH | DIALUP | LOCAL | NETWORK | REMOTE}**

  Specifies the rules under which access is to be granted or denied. If you do not specify this qualifier, the default is the type of the current process. This qualifier is usually used for detached processes (detached processes do not have a default type).

  This qualifier requires IMPERSONATE privilege.

  Note

  If you are running COM on OpenVMS Version 7.3-2 or higher, the **/TYPE=BATCH** qualifer is not supported, and the IMPERSONATE privilege is not required.

- **/WRITE_FILE** [=*file*]

  This qualifier causes the system to write the specified domain name, user account name, and password into an output file to be used later (see the **/READ_FILE** qualifier), instead of using the user-supplied password.

  If you do not provide a file specification, the system uses the following default location and file name:

  ```
  DCE$COMMON:[000000]NTA$LOGON.DAT
  ```

  Caution

  The **/READ_FILE** and **/WRITE_FILE** qualifiers are intended to be used only by servers that have no other way to acquire Windows credentials to access the OpenVMS Registry or COM for OpenVMS facilities. HP

does not recommend general use of the **/READ_FILE** and **/WRITE_FILE** qualifiers.

Once you have written a password into a disk file, HP recommends you take strong precautions to protect the password file from unauthorized access.

8.2.2 Examples of Using NTA$LOGON to Acquire Windows Credentials

Example 8–2 shows how a user acquires NT credentials for the first time.

Example 8–2 Acquiring Windows Credentials for the First Time

```
$ NTLOGON :== $NTA$LOGON
$ NTLOGON/LIST
ERROR: NtOpenProcessToken() failure: -1073741700 0xc000007c
%SYSTEM-E-NOSUCHEXT, no such extension found

$ NTLOGON/LOG JOESMITH
[Persona #1 NT extension: Account= "JOESMITH" Domain= "NT_DOMAIN" ]
Password:
```

Example 8–3 shows how the user replaces the Windows credentials.

Example 8–3 Replacing Windows Credentials

```
$ NTLOGON/DELETE
$ NTLOGON/OVERRIDE_MAPPING/DOMAIN=OTHER_DOMAIN
Username: janebrown
Password:
```

Example 8–4 shows how a user saves a password in a disk file. The system requests that the user enter the password twice with echoing suppressed.

Example 8–4 Saving a Password to a File

```
$ NTLOGON :== $NTA$LOGON
$ NTLOGON/WRITE_FILE=DEV:[DIR]NTA$LOGON.DAT COM_SERVER
Password:
Confirm:
$ NTLOGON/READ_FILE=DEV:[DIR]NTA$LOGON.DAT/LIST
File DEV:[DIR]NTA$LOGON.DAT contains the following records:
02-MAR-1999 16:57:23.20 COM_SERVER
```

After you have created this file, you can add the following to a DCL command procedure:

```
$ NTLOGON :== $NTA$LOGON
$ NTLOGON/READ_FILE=DEV:[DIR]NTA$LOGON.DAT COM_SERVER
```

8.3 The Authentication and Credential Management (ACM) Authority

The Authentication and Credential Management authority authenticates users and determines the user security profile for OpenVMS and Windows. The ACME_SERVER process provides these ACM services. The ACME_SERVER process uses plug-in modules called ACME agents. ACME agents perform the actual work of responding to authentication requests, query requests, and event requests.

The OpenVMS ACME agent (VMS$VMS_ACMESHR.EXE) provides OpenVMS native services. The MSV1_0 ACME agent (PWRK$MSV1_0_ACMESHR.EXE, an HP Advanced Server for OpenVMS product component) provides Windows connectivity services.

The MSV1_0 ACME agent forwards Windows connectivity service requests from NTA$LOGON and SSPI/NTLM to an HP Advanced Server for OpenVMS process running on one or more systems in the cluster. The PWRK$ACME_SERVER logical name can contain a comma-delimited list of cluster node names to which the MSV1_0 ACME can forward requests. Running the HP Advanced Server for OpenVMS process on more than one cluster node and including the node names in the PWRK$ACME_SERVER logical name allows the MSV1_0 ACME agent to fail over a request automatically if a connection is interrupted. If the logical name is undefined, the system defaults to the local machine name.

The ACME_SERVER process must be present on any system running RPC or COM for OpenVMS. However, the HP Advanced Server for OpenVMS process needs to be present on only one node in the cluster.

8.3.1 Windows Authentication on OpenVMS

Because the ACME_SERVER returns to its callers a complete OpenVMS persona with the requested attached Windows persona extension, the VMS ACME agent enforces the following rules:

- Every Windows user must be mapped to a local OpenVMS user name.

  The MSV1_0 ACME provides this mapping through the HP Advanced Server for OpenVMS HOSTMAP database.

- The mapped OpenVMS user name must be a valid (and not disabled) account in the SYSUAF.DAT. The account's access restrictions must allow access during the specified days and times. COM for OpenVMS and RPC typically require NETWORK access during authentication.

- The mapped OpenVMS user name must be an account with the EXTAUTH flag set. EXTAUTH allows the system manager fine control over which OpenVMS accounts can be used for mapping. You can use the IGNORE_ EXTAUTH bit (bit number 11 [decimal]) in the SECURITY_POLICY system parameter to override this per-account feature. If you set the IGNORE_ EXTAUTH bit to 1, OpenVMS allows you to map to any account, regardless of the account's EXTAUTH setting. Note that the IGNOTE_EXTAUTH is used only for the ACME_SERVER and is ignored by Loginout.

8.3.2 Managing the ACME_SERVER Process (ACME Server Commands)

To start the ACME_SERVER process and configure the MSV1_0 ACME agent at system startup, add the following entry to SYLOGICALS.COM:

```
$ DEFINE NTA$NT_ACME_TO_BE_STARTED YES
```

You can also start the ACME_SERVER process manually using the following startup command file:

```
$ @SYS$STARTUP:NTA$STARTUP_NT_ACME
```

To shut down ACME_SERVER, enter the following command:

```
$ SET SERVER ACME/EXIT
```

If an abnormal condition in an ACME agent prevents a normal server shutdown, use the **/ABORT** qualifier in the place of the **/EXIT** qualifier to force the ACME_SERVER to terminate.

To turn on ACME_SERVER logging, enter the following command:

```
$ SET SERVER ACME/LOG
```

This command creates a ACME$SERVER.LOG file in the SYS$MANAGER directory. You might find this file useful when you are trying to diagnose potential problems.

To display the ACME_SERVER configuration information, enter the following command:

```
$ SHOW SERVER ACME[/FULL]
```

8.3.3 Configuring the MSV1_0 ACME Agent

Table 8–2 lists and describes systemwide logical names you can use to control certain features of the MSV1_0 ACME agent.

Table 8–2 MSV1_0 ACME Agent Logical Names

Logical name	Description
PWRK$ACME_SERVER	Comma-delimited list of cluster SCS node names that are running HP Advanced Server for OpenVMS processes that can service Windows connectivity requests. If you do not define the node names, the MSV1_0 ACME agent tries to connect to the HP Advanced Server for OpenVMS process on the local system.
PWRK$ACME_RETRY_COUNT	The maximum number of retry attempts the MSV1_0 ACME agent performs when connecting to an HP Advanced Server for OpenVMS process. The default value is 10.
PWRK$ACME_RETRY_INTERVAL	The number of tenths of seconds between retry attempts. The default is 2.5 seconds.

9
Active Template Library

9.1 COM for OpenVMS and ATL

ATL (Active Template Library) is a set of template-based C++ classes from Microsoft that simplify the development of COM components. ATL provides support for key COM features, such as stock implementations of `IUnknown`, `IClassFactory`, `IDispatch`, dual interfaces, and connection points. It also provides support for more advanced COM features, such as enumerator classes and tear-off interfaces.

The ATL COM AppWizard and ATL Object Wizard in Microsoft Visual Studio can be used to quickly create code for simple COM objects that can be copied to OpenVMS systems and built with very few modifications.

The COM for OpenVMS ATL is based on Microsoft ATL Version 3.0. You must be running COM Version 1.1-B or higher for OpenVMS. ATL on OpenVMS Alpha requires Compaq C++ Version 6.2-016 or higher.

COM for OpenVMS provides ATL as source code in header files that you include in your application.

Table 9–1 shows the differences between the ATL implementation on Windows and OpenVMS.

Table 9–1 ATL Implementation Differences

Implementation	Windows	OpenVMS
Interface	GUI	Character cell
Server models	Single threaded or multithreaded	Multithreaded only
ATL available as DLL	Yes (not required)	No
Application registration	Automatic using UpdateRegistryFromResource function in ATLBASE.H	Automatic using UpdateRegistryFromFile function in ATLBASE.H
ATL component types	In process as DLL Out of process as EXE	In process as shareable image Out of process as an executable image

9.2 Developing a COM for OpenVMS Application Using ATL

The following sections describe how to create a COM for OpenVMS application using ATL.

9.2.1 Step 1: Create the ATL Component in Microsoft Visual Studio

Generate the code using the Microsoft Visual Studio ATL COM AppWizard. For information about using the ATL COM AppWizard, see the Microsoft Developer Network (MSDN) documentation.

Copy the generated files to OpenVMS. For example, copy the files using File Transfer Protocol (FTP) in ASCII mode. Table 9–2 lists and describes the files that the ATL COM AppWizard would generate for a project named `mycomapp`.

Table 9–2 Files Generated by ATL COM AppWizard for `mycomapp`

File name	Description	Platform	In Process or Out of Process
`mycomapp.cpp`	Contains the implementation of DllMain, DllCanUnloadNow, DllGetClassObject, DllRegisterServer and DllUnregisterServer. Also contains the object map, which is a list of the ATL objects in your `mycomapp`. This is initially blank, because you have not created an object yet.	Windows/OpenVMS	Both
`mycomapp.def`	The standard Windows module definition file for the DLL. **Note:** `MYCOMAPP.DEF` becomes `MYCOMPAP$SHR.OPT` on OpenVMS.	Windows	In process
`mycomapp.dsw`	The `mycomapp` workspace.	Windows	Both
`mycomapp.dsp`	The file that contains the `mycomapp` settings.	Windows	Both
`mycomapp.idl`	The interface definition language file, which describes the interfaces specific to your objects.	Windows/OpenVMS	Both
`mycomapp.rc`	The resource file, which initially contains the version information and a string containing the `mycomapp` name.	Windows	Both
`Resource.h`	The header file for the resource file.	Windows/OpenVMS	Both
`mycomappps.mk`	The make file that can be used to build a proxy/stub DLL. You do not need this file.	Windows	Proxy/stub
`mycomapps.def`	The module definition file for the proxy/stub DLL. **Note:** `MYCOMAPPPS.DEF` becomes `MYCOMAPPPS$SHR.OPT` on OpenVMS.	Windows	Proxy/stub
`StdAfx.cpp`	The file that will include the ATL implementation files.	Windows/OpenVMS	Both
`StdAfx.h`	The file that will include the ATL header files. To make the `mycomapp` DLL useful, you need to add a control, using the ATL Object Wizard.	Windows/OpenVMS	Both
`mycomapp.rgs`	A registrar script for your COM server.	Windows/OpenVMS	Both
`myinterface.rgs`	A registrar script for your COM server.	Windows/OpenVMS	Both
`myinterface.cpp`	The interfaces specific to your object.	Windows/OpenVMS	Both

(continued on next page)

Table 9–2 (Cont.) Files Generated by ATL COM AppWizard for `mycomapp`

File name	Description	Platform	In Process or Out of Process
`myinterface.h`	The header file for the interfaces.	Windows/OpenVMS	Both

9.2.2 Step 2: Modify Generated Files for ATL Applications on OpenVMS

Make the following changes to the generated files before you build ATL applications on OpenVMS.

9.2.2.1 Remove _ATL_MIN_CRT

When the ATL COM AppWizard generates `mycomapp`, it also defines the macro `_ATL_MIN_CRT` as part of the GUI support. Because OpenVMS does not have a graphical interface, you must remove (or not define) `_ATL_MIN_CRT` when you build on OpenVMS.

9.2.2.2 Include ATLMAIN.CXX

On OpenVMS, you must include ATLMAIN.CXX for out of process components. ATLMAIN.CXX defines the `wWinMain()` function.

9.2.2.3 Modify Registration Procedure

OpenVMS does not support registering the application using the `UpdateRegistryFromResource` function; rather, you must use the OpenVMS `UpdateRegistryFromFile` function in the `ATLBASE.H` header file. You must make the following changes to your application:

File to search	Search for	Replace with
Interface header file	DECLARE_REGISTRY_RESOURCEID	DECLARE_REGISTRY_FILE
Project source file	_Module.UpdateRegistryFromResource	_Module.UpdateRegistryFromFile

The following example shows sample coding changes:

```
#ifdef __vms
DECLARE_REGISTRY_FILE(_T("MYINTERFACE.RGS"))
#else
DECLARE_REGISTRY_RESOURCEID(IDR_MYINTERFACE)
#endif
```

```
#ifdef __vms
_Module.UpdateRegistryFromFile(_T_"MYCOMAPP.RGS"), TRUE);
#else
_Module.UpdateRegistryFromResource(IDR_MYCOMPAPP, TRUE);
#endif
```

9.2.2.4 Remove Calls to Windows Message Functions for OpenVMS V7.3-2

Beginning in OpenVMS Version 7.3-2, HP no longer supports the PostThreadMesssage, GetMessage and DispatchMessage calls that are generated by the ATL COM AppWizard for out of process components.

The same functionality can be achieved by replacing these calls with the event-handling functions CreateEvent, SetEvent, and WaitForSingleObject. Refer to the source code for TESTATL.CXX in DCOM$EXAMPLES:[TESTATL_OUTPROC] for more information.

The following example shows some of the sample coding changes

```
#ifdef __vms
  SetEvent(hMsg);
#else
  PostThreadMessage(dwThreadID, WM_QUIT, 0, 0);
#endif
```

```
#ifdef __vms
  WaitForSingleObject(hMsg, INFINITE);
#else
  MSG msg;
  while (GetMessage(&msg, 0, 0, 0))
        DispatchMessage(&msg);
#endif
```

9.2.3 Step 3: Build an Application Using the MIDL Compiler

This process is the same as the one shown in Section 7.2.

In-process example:

```
$ MIDL :== $DCOM$MIDL.EXE
$ MIDL  -nologo -Oicf mycompapp.idl -
  -IDCOM$LIBRARY -
  -iid mycompapp_i.cxx -
  -proxy mycompapp_p.cxx -
  -dlldata dlldata.cxx -
  -tlb mycompapp$shr.tlb
```

Out-of-process example:

```
$ MIDL :== $DCOM$MIDL.EXE
$ MIDL  -nologo -Oicf mycompapp.idl -
  -IDCOM$LIBRARY -
  -iid mycompapp_i.cxx -
  -proxy mycompapp_p.cxx -
  -dlldata dlldata.cxx -
  -tlb mycompapp.tlb
```

HP recommends that the name of your type library match the name of your executable or shareable image.

9.2.4 Step 4: Compile the ATL COM Application

The following sections describe how to compile COM for OpenVMS applications.

9.2.4.1 Required Header File: ATLBASE.H

The `VMS_ATL.H` header file defines several macros used by the header files. `VMS_ATL.H` is already included in the `ATLBASE.H` header file. When you create ATL source code, you must include `ATLBASE.H` as the first noncommented line in your source (both header and implementation) files.

9.2.4.2 Required Macro Definitions

Include the following /DEFINE qualifier on all of your CXX commands:

```
/DEFINE=(UNICODE=1,_WIN32_DCOM,_ATL_STATIC_REGISTRY)
```

The UNICODE macro ensures that wide-character variants of Win32 APIs and data structures are enabled when you compile. (The UNICODE macro is also defined in VMS_DCOM.H.)

The _ATL_STATIC_REGISTRY macro enables you to statically link with the ATL registry component (Registrar) for optimized registry access. You can add the macro either by including the /DEFINE qualifier on the command line or by adding the stdafx.h header file to your code.

9.2.4.3 Required Include Directories

COM for OpenVMS applications typically require header files that come from DCOM$LIBRARY. The ATL header files and source files are also located in DCOM$LIBRARY.

Include the following qualifier on your CXX command lines:

```
/INCLUDE=DCOM$LIBRARY
```

If you already have an /INCLUDE qualifier on your command line, modify the command to include DCOM$LIBRARY.

9.2.4.4 Required C++ Qualifiers

You must specify the following C++ qualifiers when you build COM for OpenVMS applications:

- /EXCEPTIONS=CLEANUP

  Specify the /EXCEPTIONS=CLEANUP qualifier on C++ commands to enable C++ exceptions.

- /STANDARD=MS

  The C++ compiler supports many different compilation standards. You must use /STANDARD=MS for COM applications created by ATL.

  /STANDARD=MS informs the compiler that it should follow the language constructs supported by the Visual C++ compiler. This switch works well with the code generated by ATL.

- /TEMPLATE_DEFINE=(NOALL,NOPRAGMA)

  This switch controls the instantiation of C++ templates. You must specify the following options:

  — [NO]ALL

  Instantiate all function template entities declared or referenced in the compilation unit, including typedefs. For each fully instantiated template class, all its member functions and static data members are instantiated even if they were not used. Nonmember template functions are instantiated even if the only reference was a declaration. Instantiations are created with external linkage. Overrides /REPOSITORY at compile time. The compiler places instantiations in the user's object file. The template definition must be present before the point of each instantiation in the source file.

  The default is /TEMPLATE_DEFINE=NOALL.

  — [NO]PRAGMA

  Determines whether the C++ compiler ignores #PRAGMA DEFINE_TEMPLATE directives encountered during the compilation. This option lets you quickly switch to automatic instantiation without having to remove all the pragma directives from your program's code base.

  The default is /TEMPLATE_DEFINE=PRAGMA, which enables #PRAGMA DEFINE_TEMPLATE.

9.2.5 Step 5: Link the ATL COM Application

To build a COM for OpenVMS application, you must build both client and component images. Because you can implement a component as either an in process component or an out of process component, you must build a shareable image or an executable image, or both.

The following sections describe the steps you must follow to link the client, component, and proxy/stub images.

9.2.5.1 Linking the Client and the Out of Process Component

Although you do not need to specify any qualifiers to link the client or the component executable images, you must link both images. The specific link-time dependency is as follows:

- `DCOM$LIBRARY:DCOM.OPT`

If you have one or more C++ modules, use the C++ linker (CXXLINK) instead of the standard OpenVMS linker so you can specify the location of your C++ repository (/CXX_REPOSITORY qualifier). For example:

```
$  CXXLINK/your-specific-linker-qualifiers list-of-object-modules, -
_$ DCOM$LIBRARY:DCOM.OPT/OPTIONS, application.OPT/OPTIONS  -
_$ /REPOSITORY=[.CXX_REPOSITORY]
```

You can also include the list of object modules in an options file instead of on the command line.

9.2.5.2 Linking the In Process Component Shareable Image

The in process component shareable image dependency list differs slightly from that of the client and component executables. The specific link-time dependencies are as follows:

- `[`*directory-name*`]MYCOMPAPP$SHR.OPT`
- `DCOM$LIBRARY:DCOM.OPT`

9.2.5.3 Creating a Symbol Vector

To create a symbol vector for the in process component shareable image, use the procedure described in Section 7.4.2.1.

To create a symbol vector for the proxy/stub shareable image, use the procedure described in Section 7.4.3.

9.3 ATL Samples

TESTATL is an out of process sample, and MATH101 is an in process sample.

You can find the sample ATL applications shown in this chapter in the following directories on the COM for OpenVMS kit:

```
DCOM$EXAMPLES:[TESTATL_OUTPROC]
DCOM$EXAMPLES:[TESTATL_INPROC]
```

Note

If you are running authenticated COM, before you build the application on OpenVMS you must run NTA$LOGON and acquire Windows credentials. For more information, see Section 8.2.

9.3.1 Out of Process COM Sample (TESTATL_OUTPROC)

This sample implements a COM client and server in which the component provides one interface: `ISum`.

Given sources initially generated by the Microsoft Visual Studio ATL AppWizard and a few applied changes, the sample demonstrates the build, registration, and execution of the ATL application on OpenVMS.

The following sections describe how to create the application using the Microsoft ATL AppWizard on Windows and how to build the application on an OpenVMS system.

9.3.1.1 Creating the Application on Windows

To generate a skeleton project and simple objects using the Microsoft Visual Studio ATL AppWizard, follow these steps:

1. Generate the skeleton project:
   - Select the ATL COM AppWizard and name your skeleton project.
   - Choose **Executable (EXE)** as the server type.
2. Add objects:
   - Start the ATL Object Wizard.
   - From the **Objects** category, select **Simple Object**.
   - From the **Attribute** tab, choose the **Both** threading model.

9.3.1.2 Building, Registering, and Running the Application on OpenVMS

A README file describes how to build, register, and run this COM for OpenVMS sample. The file is located in:

```
DCOM$EXAMPLES:[TESTATL_OUTPROC]README-TESTATL_OUTPROC.TXT
```

9.3.2 In-Process COM Sample (TESTATL_INPROC)

This sample implements a COM client and server in which the component provides three interfaces: `ISum`, `IDiv`, and `IMul`.

Given sources initially generated by the Microsoft Visual Studio ATL AppWizard, the sample demonstrates the build, registration, and execution of the shareable application on an OpenVMS system.

The following sections describe how to build the application.

9.3.2.1 Creating the Application on Windows

To generate a skeleton project and simple objects using the Microsoft Visual Studio ATL AppWizard, follow these steps:

- Generate the skeleton project:
  - Select the ATL COM AppWizard and name your skeleton project.
  - Choose **Dynamic Link Library (DLL)** as the server type.
- Add objects:
  - Start the ATL Object Wizard.
  - From the **Objects** category, select **Simple Object**.
  - From the **Attribute** tab, choose the **Both** threading model.

9.3.2.2 Building, Registering, and Running the Application on OpenVMS

A README file describes how to build, register, and run this COM for OpenVMS sample. The file is located in:

```
DCOM$EXAMPLES:[TESTATL_INPROC]README-TESTATL_INPROC.TXT
```

9.4 Suggested Reading

The following resources provide more information about ATL:

- Third-party books about ATL:
  - — *Beginning ATL COM Programming*, Grimes and Stockton, Templeman and Reilly, Wrox Press, Olton, Birmingham, UK, 1998. ISBN: 1-861000-11-1.
  - — *Professional ATL COM Programming*, Dr Richard Grimes, Wrox Press, Olton, Birmingham, UK, 1998. ISBN: 1-861001-4-01.
- Websites:
  - — *The Component Object Model Specification*, available from the Microsoft COM website:

    ```
    www.microsoft.com/com
    ```

10

COM for OpenVMS and DLL Surrogates

COM for OpenVMS makes it possible to create an in-process server that can be loaded into a surrogate process. The default surrogate provided, DCOM$DLLHOST.EXE, can be run remotely and instructed to load any in-process component, providing it with a surrogate parent process and security context.

Running an in-process server in a surrogate process offers several possible benefits:

- Provides fault isolation and the ability to service multiple clients simultaneously.
- In a distributed environment, a DLL server implementation can service remote clients.
- Permits clients to take advantage of services the in-process server provides, while protecting themselves from untrusted components.
- The server is provided with the security context of the surrogate process.

10.1 Running Your Components in the Context of a DLL Surrogate

To run your application using a Dllhost Surrogate, you need to set some values in your OpenVMS Registry.

Add the following values to the OpenVMS Registry:

```
[HKEY_CLASSES_ROOT\CLSID\{Guid}]
"AppID"="{Guid}"

[HKEY_CLASSES_ROOT\CLSID\{Guid}]\InProcServer32
"ThreadingModel"="Free"

[HKEY_CLASSES_ROOT\APPID\{Guid}]
"DllSurrogate"=""
```

The registry code in the DLL Surrogate sample (REG_SURROGATE.CXX) includes code that makes these changes.

In addition, if you plan to run multiple clients launched by different users within the same surrogate process, you need to change one of the application properties. Specifically, you must set a RunAs account (NTLM account) through DCOM$CNFG.

To set a RunAs account, perform the following steps:

1. Start the DCOM$CNFG utility (see Section 6.3).
2. Select option 1—Applications list.
3. Enter a number for the application you want to change.
4. Select option 3—Identity.

5. Modify the Application Identity (see Section 6.3.4).

By default, the Identity on an application is set as Launching User. This causes a separate surrogate process to be launched for each different user.

To run legacy applications within a surrogate, you do not need to modify any code. Add the preceding values to the OpenVMS Registry, and **delete** the following LocalServer32 key:

```
[HKEY_CLASSES_ROOT\CLSID\{Guid}]\LocalServer32
```

10.2 Developing a Surrogate Application

For more information about how to run your application within a surrogate and configure your Registry values, see the DLL Surrogate sample in the DCOM$EXAMPLES:[SURROGATE] directory in the COM for OpenVMS kit.

11

COM for OpenVMS and IEEE Floating Point

COM for OpenVMS supports IEEE floating-point values in COM for OpenVMS applications.

11.1 Running Sample Programs with IEEE Floating Point Values

You can run any sample program with IEEE floating-point values rather than VAX floating-point values. To do so, apply the following changes to your application:

- Add the following API below any included header files:

  ```
  VMS$UseIeeeFloatingPoint();
  ```

- Add the following switch to the compile statement for the sample to which you added the API:

  ```
  /FLOAT=IEEE_FLOAT
  ```

11.2 Restrictions Using IEEE Floating-Point Values in COM for OpenVMS Applications

IEEE floating-point values can be used in COM for OpenVMS applications running between remote OpenVMS systems, running between Windows and OpenVMS, and running an out-of-process server on OpenVMS.

IEEE floating-point values cannot be used in the following circumstances:

- With an in-process server
- With another sample not compiled with the IEEE qualifier

Part II

OpenVMS Registry

The following chapters describe the OpenVMS Registry database and its structure.

The OpenVMS Registry $REGISTRY and $REGISTRYW system services are described in the *OpenVMS System Services Reference Manual*.

For the latest information about the OpenVMS Registry, refer to the *OpenVMS Release Notes* for the current version of the operating system.

12
Overview of OpenVMS Registry

12.1 What is the Registry?

The Windows Registry is a single, systemwide, hierarchical database of configuration information about hardware and software (both the operating system and applications). The Windows Registry replaced Windows 3.*x* `.ini` files, providing a single place for storing application and configuration information.

To allow OpenVMS and Windows to interoperate, HP has provided a registry on OpenVMS. Like the Windows Registry, the OpenVMS Registry is made up of two components: the OpenVMS Registry database and the OpenVMS Registry server. The OpenVMS Registry database is a systemwide or clusterwide hierarchical database of configuration information. This information is stored in a database structure of keys and associated values. The OpenVMS Registry server controls all OpenVMS Registry operations, such as creating and backing up the OpenVMS Registry database, and creating, displaying, modifying, or deleting keys and values.

The OpenVMS Registry includes interfaces (COM APIs and system services) to allow applications to control the OpenVMS Registry server and to read and write to the OpenVMS Registry database. The OpenVMS Registry also includes server management utilities to allow system managers to display and update OpenVMS Registry information from the OpenVMS DCL command line.

The OpenVMS Registry is compatible with the Windows Registry. Windows client applications such as `RegEdt32` can connect to and edit the OpenVMS Registry.

12.1.1 Suggested Reading

The following resources can provide you with more information about Windows Registry and related topics:

- Third-party books about the Windows Registry:
  - *Windows Server NT 4.0 Unleashed*, Jason Garms, SAMS Publishing, Indianapolis, IN, 1998. ISBN: 0-672-30933-5.

12.2 OpenVMS Registry Concepts and Definitions

The OpenVMS Registry, like the Windows Registry, is a hierarchical database with several branches.

The following sections list and explain OpenVMS Registry database elements and operation.

12.2.1 Keys, Subkeys, and Values

A **key** is one of the basic building blocks of the OpenVMS Registry database. A key contains information specific to the computer, system, or user; it is a header field in the OpenVMS Registry database. Keys can be arranged in a hierarchy (or tree).

There are two main (or root) keys in the OpenVMS Registry:

- `HKEY_USERS` contains information about each user.
- `HKEY_LOCAL_MACHINE` contains hardware, software, security, and general system configuration information.

The key `HKEY_CLASSES_ROOT` points to the `CLASSES` subkey in `HKEY_LOCAL_MACHINE`. These root keys are discussed in more detail in Section 12.3.

A **subkey** is a key that is a child to another key. A key can have zero or more subkeys. Subkeys allow you to group related keys together below another key in a hierarchy or tree.

A **value entry** (or **value**) is a named element of data; it is a record field in the registry database. A key has zero or more associated values. A value has a value name, a value type, a collection of flags, and associated data (defined by the value's type). OpenVMS Registry supports the following value types:

- Null-terminated string
- Null-terminated array of null terminated strings
- Null-terminated string containing environment variables (logical names or symbols)
- 32-bit data item
- 64-bit data item
- Raw binary

Figure 12–1 summarizes the relationship between keys, subkeys, and values.

Figure 12–1 Key, Subkey, and Value Relationships

```
Key1=Value1
Key2
  |
  +-Subkey1=Value1
  |
  +-Subkey2=Value1,Value2
  :
  .
```

12.2.1.1 Key and Value Volatility

You can define OpenVMS Registry keys and values as either **nonvolatile** or **volatile**. Nonvolatile keys are saved to OpenVMS Registry files. Volatile keys are cached to a temporary file.

On Windows systems, volatile keys and values are removed when the system restarts.

On OpenVMS, volatile keys and values are automatically removed when all nodes in a cluster are rebooted. OpenVMS extends the lifetime of volatile keys to survive server failover but not a cluster reboot. (In a standalone system, volatile keys and values are lost when the system reboots.)

12.2.1.2 Key Write-through and Write-behind

When you create a key, you can specify when the OpenVMS Registry should write that key's changed information. The write options are as follows:

- **Write-through:** Write the changes to disk immediately.
- **Write-behind:** Cache the changes and write them later.

The `Cache Action` attribute allows you to specify a key's write characteristics. If you do not specify the cache action attribute when you create the key, the key inherits this attribute from its parent.

When you use the `SYS$REGISTRY` interface, you can use the the `REG$M_NOW` function code modifier for a request in progress to force an immediate write (write-through), regardless of the cache action attribute value.

12.2.1.3 Linking a Key to Other Keys and Values

OpenVMS Registry keys can link to other OpenVMS Registry keys, providing multiple paths to the same piece of data. In the same way, OpenVMS Registry values can link to other OpenVMS Registry values. These key and value links, or **symbolic links**, are similar to file links. Symbolic links are name references.

For example, you can link `Key A` to `Key B`. When you query `Key A` and its value, the system returns `Key B`'s value.

You can also **chain** symbolic links. That is, `Key A` can point to `Key B` and `Key B` can point to `Key C`; as a result, `Key A` also points to `Key C`. You can specify a link through the $REGISTRY system service or through the OpenVMS Registry server management command-line interface.

12.2.1.4 Rules for Creating OpenVMS Registry Keys and Value Names

The following rules apply to key and value names:

- A key can have subkeys and values.
- A key name can be composed of any Unicode (4 bytes) character except the backslash (`\`) character and the null character. You must specify at least one character.
- A value name can be composed of any Unicode (4 bytes) character.
- A key string can be either a name (for example, `disk`) or a path (for example, `Hardware\cosmos\disk`).
- A value string can be a name *only*.
- When you define a key, if you specify a path but the system does not find one or more of the path subkeys, the system creates these subkeys automatically. The created keys inherit the attributes of their parent.
- The key and value names are case preserved in the OpenVMS Registry database. Name comparisons are case insensitive unless you specify the `REG$M_CASESENSITIVE` function code modifier with calls to the $REGISTRY system service.

- For pure binary data, the maximum size of a value is 1 MB (for Windows compatibility).

12.2.2 Class

The `Class` attribute allows you to store additional descriptive information with each key. For example, specifying `Class text string` could allow you store permitted data types with a specified key.

12.2.3 Hive

A **hive** is a collection of related keys, subkeys, and values stored in the OpenVMS Registry.

On Windows systems, a hive is stored in a single file in the `%SystemRoot%\system32\config` directory, along with an associated `LOG` file. Windows allows users to save hives to specified files on disk so that these files can be loaded at a later time.

On OpenVMS systems, the entire OpenVMS Registry database consists of two hives: `REGISTRY$LOCAL_MACHINE.REG` and `REGISTRY$USERS.REG`. OpenVMS does not support loading and unloading hives.

12.3 OpenVMS Registry Structure

To allow Windows applications to interface with the OpenVMS Registry database, the OpenVMS Registry database includes a subset of the Windows Registry predefined keys and subkeys.

The OpenVMS Registry includes the following predefined standard keys:

- `HKEY_CLASSES_ROOT`

  On Windows systems, this key is reserved for the definition of classes of documents and the properties associated with these classes.

  On OpenVMS systems, this key by default does not have any subkey or value.

  This entry point maps to the `HKEY_LOCAL_MACHINE\SOFTWARE\Classes` subkey.

- `HKEY_USERS`

  On Windows systems, the entries under this entry point define the default user configuration for users on the local system and the user configuration for the current user.

  On OpenVMS systems, this key by default does not have any subkey or value.

- `HKEY_LOCAL_MACHINE`

  The entries under this entry point are reserved for system configuration information.

  On Windows systems, this area contains information about the bus type, system memory, and installed hardware and software.

  On OpenVMS systems, this key has the following predefined subkeys:

  — `Hardware`

    On Windows systems, the system constructs the volatile subkeys of this key from the information gathered at boot time.

    On OpenVMS systems, this key does not have any subkey or value by default.

  — `Security`

On Windows systems, these keys contain all the security information for the local computer. The system owns the information in these keys and protects them accordingly.

On OpenVMS systems, this key by default does not have any subkey or value.

— Software

On Windows systems, this key contains information about the software on the local system that is independent of per-user configurations.

On OpenVMS systems, this key has the following predefined subkeys:

* Classes
* Hewlett-Packard Company
* Microsoft

— System

On Windows systems, this key contains information about devices and services.

On OpenVMS systems, this key has the following predefined subkeys:

* CurrentControlSet

  On Windows systems, this key contains information about Control, Enum, and Hardware Profiles and Services.

  On OpenVMS systems, this key is reserved for use by HP Advanced Server for OpenVMS.

* Registry

  This subkey does not exist on Windows systems. On OpenVMS systems, this key contains the OpenVMS Registry server configuration parameters in the form of subkeys and values. The predefined subkeys are as follows:

  + File Quotas

    This subkey is empty when you create the OpenVMS Registry database. A system manager can assign quota for each OpenVMS Registry database file by creating a value whose name is the name of the OpenVMS Registry file. If no value exists for a file, the OpenVMS Registry server uses the default value for the **Default File Quota** setting.

    For example, a system manager could use REG$CP to assign a 1 MB quota to the OpenVMS Registry REGISTRY$LOCAL_MACHINE.REG file by issuing the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE VALUE/NAME=REGISTRY$LOCAL_MACHINE/TYPE=DWORD/ -
_$ DATA=%D1000000 "hkey_local_machine\system\registry\File Quotas"
```

  + File Monitor

    This subkey is not used.

  + Priority

This subkey is empty at OpenVMS Registry database creation. A system manager can change the priority of the OpenVMS Registry server on a specified node by creating a value whose name is the node name of the system in the cluster on which the OpenVMS Registry server resides.

For example, a system manager could use the REG$CP server management utility to assign a priority of 100 to node COSMOS by issuing the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE VALUE/NAME=COSMOS/TYPE=DWORD/DATA=%D100 -
_$ "hkey_local_machine\system\registry -
_$ \Priority"
```

12.4 OpenVMS Registry Restrictions and Limitations

This section contains the current restrictions and limitations in the OpenVMS Registry.

12.4.1 Registry Data Transfer Size Restriction Eased

Versions of OpenVMS prior to Version 7.3 placed restrictions on the size of a data transfer between the $REGISTRY system service and the OpenVMS Registry server. The data transfer restrictions, in turn, placed restrictions on the maximum size of a single block of data that can be stored or retrieved from the Registry database. They also limited the depth of a REG$CP Search command, and placed limits on the number of Advanced Server domain groups of which a user can be a member. These restrictions were eased in OpenVMS Version 7.3, but have not been eliminated entirely.

Previously the restrictions were approximately 8K bytes transmit (service to server) and approximately 4K bytes receive. The current restriction depends on the setting of the system parameter MAXBUF. The range for MAXBUF is 4K to 64K, with a default of 8K.

MAXBUF is the maximum allowable size for any single buffered I/O packet. You should be aware that by changing MAXBUF you also affect other areas of the system that perform buffered I/O.

12.5 Reading and Writing to the OpenVMS Registry

You can read and write to the OpenVMS Registry in the following ways:

- Using COM for OpenVMS, through the COM APIs available on OpenVMS. This allows application programmers to enter, modify, and delete OpenVMS Registry keys and values.
- Through the $REGISTRY and $REGISTRYW system services and the OpenVMS Registry server management utility commands. This allows application programmers to enter, modify, and delete OpenVMS Registry keys and values.
- From Windows, through the Windows Registry APIs, or using RegEdt32 (the Windows Registry Editor). This allows Windows users to view and edit OpenVMS Registry keys and values.

12.5.1 $REGISTRY System Services

The OpenVMS Registry includes two OpenVMS system services that provide an interface to the OpenVMS Registry server. The OpenVMS Registry system services allow you to query, update, and create keys, subkeys, and values in the OpenVMS Registry database.

For more information about the $REGISTRY and $REGISTRYW system services, see the *OpenVMS System Services Reference Manual*.

12.5.2 REG$CP Server Management Utility

The `REG$CP` server management utility allows you to display and update OpenVMS Registry information from the OpenVMS DCL prompt. The utility also allows you to back up and restore the entire OpenVMS Registry database to or from a file, as long as you have the required system privileges.

For more information about the `REG$CP` server management utility, see Chapter 14.

12.6 OpenVMS Registry Security

The OpenVMS Registry implements both the OpenVMS and Windows security models.

To access to the OpenVMS Registry database, the calling process must have the proper OpenVMS Registry rights identifier for the operation you want to perform (for example, `REG$LOOKUP` for read operations, `REG$UPDATE` for write operations, or `REG$PERFORMANCE` for statistics operations) or the calling process must have the SYSPRV privilege.

The following sections describe the two models.

12.6.1 OpenVMS Security Model

When a user requests access to the OpenVMS Registry, the OpenVMS system checks the following:

1. Does the user have Windows credentials?

   If the user has Windows credentials, OpenVMS allows the user access to the OpenVMS Registry based on the user's supplied credentials. The user can acquire Windows credentials through the following methods:

   - Connecting to the OpenVMS Registry from a Windows system
   - Running a COM for OpenVMS client
   - Running an OpenVMS SYS$ACM client that acquires NT credentials

   If the user is not allowed access to the OpenVMS Registry based on the user's supplied credentials, or if the user does not have Windows credentials, continue to the next step.

2. Does the user have the OpenVMS SYSPRV privilege?

   - If the user has the SYSPRV privilege, OpenVMS allows the user full access to the OpenVMS Registry.
   - If the user does not have the SYSPRV privilege, continue to the next step.

3. Does the user have the REG$UPDATE, REG$LOOKUP, or REG$PERFORMANCE rights identifier?
   - If the user has the REG$UPDATE, REG$LOOKUP, or REG$PERFORMANCE rights identifier, OpenVMS allows the user access to the OpenVMS Registry using the supplied rights identifier. The user can access the OpenVMS Registry database as follows:
     — REG$UPDATE: Allows full access to the OpenVMS Registry except for maintenance requests.
     — REG$LOOKUP: Allows read-only access to the OpenVMS Registry.
     — REG$PERFORMANCE: Allows access to performance data collected by the OpenVMS Registry server.
   - If the user does not have the REG$UPDATE, REG$LOOKUP, or REG$PERFORMANCE rights identifier, continue to the next step.
4. If the user has no Windows credentials, OpenVMS grants the OpenVMS user Windows Everyone group access. In this case, the OpenVMS user's access to OpenVMS Registry keys depends on what permissions the key owner defined for Everyone when the key owner created the key or subkey. Based on these permissions, the OpenVMS user will be able to do one of the following:
   — Read the key and its subkeys.
   — Not see the key and its subkeys.

12.6.1.1 Granting OpenVMS Registry Access Rights Using the AUTHORIZE Utility

You can use the OpenVMS Authorize utility (AUTHORIZE) to add the SYSPRV privilege and REG$UPDATE, REG$LOOKUP, and REG$PERFORMANCE identifiers to users.

Caution

Granting OpenVMS Registry rights overrides Windows security access checks.

Because rights identifiers are specific to an application, you cannot use the AUTHORIZE command to create the rights identifiers. Use the REG$CP server management utility to create these rights identifiers on your system. Running the REG$CP server management utility creates these rights by default. You must run REG$CP from a privileged account. For more information about running REG$CP, see Chapter 14.

The following example shows how to use the SET RIGHTS_LIST command to allow all users to view keys and data in the OpenVMS Registry database. This command adds the REG$LOOKUP identifier to the system rights list.

```
$ SET RIGHTS_LIST/ENABLE/SYSTEM REG$LOOKUP
```

Example 12–1 shows how to use AUTHORIZE to grant and remove OpenVMS Registry rights to a specific user.

Example 12–1 Using AUTHORIZE to Grant Rights to a User

(continued on next page)

Example 12–1 (Cont.) Using AUTHORIZE to Grant Rights to a User

```
$ SET DEF SYS$SYSTEM
$ RUN AUTHORIZE

UAF> GRANT/IDENTIFIER REG$LOOKUP SMITH ❶
UAF> GRANT/IDENTIFIER/ATTRIBUTES=DYNAMIC REG$UPDATE SMITH ❷
UAF> REVOKE/IDENTIFIER REG$UPDATE SMITH ❸
UAF> GRANT/IDENTIFIER REG$PERFORMANCE SYSTEM ❹
```

❶ This AUTHORIZE command grants the REG$LOOKUP identifier to user Smith, allowing Smith to view keys and data in the OpenVMS Registry database.

❷ This AUTHORIZE command grants the REG$UPDATE identifier to user Smith, allowing Smith to modify keys and data in the OpenVMS Registry database. The dynamic attribute allows Smith to remove or restore the REG$UPDATE identifier from the process rights list by using the SET RIGHT/ENABLE or the SET RIGHT/DISABLE command.

❸ This AUTHORIZE command removes the REG$UPDATE identifier from user Smith.

❹ This AUTHORIZE command grants the REG$PERFORMANCE identifier to the system manager account, allowing the system manager to enable and disable the monitoring of OpenVMS Registry performance data.

12.6.2 Windows Security Model

Windows users can access the OpenVMS Registry only through the HP Advanced Server for OpenVMS. OpenVMS grants Windows users access to the OpenVMS Registry based on the user's Windows credentials.

12.7 Controlling the OpenVMS Registry Server Operations

OpenVMS Registry server operations include control of file quotas, server priority, error recovery actions, frequency of database backup, and OpenVMS Registry server tuning.

The following sections describe OpenVMS Registry server operations, and provide minimum, maximum, and default values for each setting. For information about how to change these settings, see Chapter 14.

12.7.1 Defining Maximum Reply Age/Age Checker Interval Settings

The OpenVMS Registry server handles duplicate requests by tracking work in progress and returning a REG$_DUPLREQUEST error. The OpenVMS Registry server also holds completed requests in case a duplicate request is received for work that is already completed. In this case, the OpenVMS Registry server reconstructs the reply. After a specified time, the requests are discarded. The **Maximum Reply Age** setting determines how long these requests are retained. The **Age Checker Interval** setting determines how often the OpenVMS Registry server checks for requests that exceed this age.

By default, the server checks for old completed requests every five seconds. By default, the server discards completed requests that are older than five seconds.

Setting Name	Default value	Minimum value	Maximum value
Maximum Reply Age	5	1	60
Age Checker Interval	5	1	60

12.7.2 Defining the Database Log Cleaner Interval/Initial Log File Size Settings

The OpenVMS Registry uses a a two-phase commit process to write modifications to the OpenVMS Registry database. The OpenVMS Registry first writes the modifications to a log file and then applies the log file to the OpenVMS Registry database. The **Database Log Cleaner Interval** setting determines how often the OpenVMS Registry applies the log file to the OpenVMS Registry database. After the OpenVMS Registry applies the log file, the OpenVMS Registry creates a new log file based on the size you specify in the **Initial Log File Size** setting.

The **Database Log Cleaner Interval** setting should be short enough so that writes to the database do not require that the log file be extended. Also, the log file size should be small to keep the amount of time spent applying the log relatively short, because this operation blocks writes to the database.

By default, the log file is applied every five seconds. By default, the OpenVMS Registry log file is created using a size of 32 blocks (16 KB).

Setting Name	Default value	Minimum value	Maximum value
Database Log Cleaner Interval	5	1	30
Initial Log File Size	32	16	256

12.7.3 Defining Default File Quota/File Quota Interval Settings

The OpenVMS Registry server limits the size of OpenVMS Registry database files by applying file quotas. You can assign file quotas to the individual files that make up the OpenVMS Registry database. If you do not assign a file quota, the OpenVMS Registry uses the **Default File Quota** setting.

Note

The **File Quota Interval** setting is used by Registry servers prior to OpenVMS V7.3-1 only.

The OpenVMS Registry server periodically recalculates the size of the OpenVMS Registry database files to see whether quota is exceeded. The **File Quota Interval** setting determines how often the OpenVMS Registry performs this calculation.

By default, the **Default File Quota** setting is 10 MB. By default, the **File Quota Interval** setting is 30 seconds.

Setting Name	Default value	Minimum value	Maximum value
Default File Quota	0x10000000	0x7d00	0x3fffffff
File Quota Interval	30	10	60

12.7.4 Defining the Scan Interval Setting

Note

The **Scan Interval** setting is used by Registry servers prior to OpenVMS Version 7.3-1 only.

In an OpenVMS Cluster, you can run OpenVMS Registry servers on more than one node; however, only one OpenVMS Registry server is active at a time. A OpenVMS Registry server's priority relative to the other OpenVMS Registry servers in the cluster determines which OpenVMS Registry server is active. If the cluster configuration changes, the system manager can adjust the priority of one or more OpenVMS Registry servers. After the system manager changes the priority, the OpenVMS Registry servers in the cluster determine which server now has the highest priority and automatically change their states as necessary. The **Scan Interval** setting determines how often a OpenVMS Registry server checks for changes in its priority.

By default, a server checks for changes in priority every 120 seconds.

Setting Name	Default value	Minimum value	Maximum value
Scan Interval	120	60	300

12.7.5 Defining the Log Registry Value Error Setting

The OpenVMS Registry server logs an error if one of the OpenVMS Registry server parameter values is out of the acceptable range. If the OpenVMS Registry detects an out-of-range error, the OpenVMS Registry server uses the default value for that parameter. The **Log Registry Value Error** setting is a Boolean value that determines whether the error should be logged.

By default, the OpenVMS Registry server does not log out-of-range errors.

Setting Name	Default value	Minimum value	Maximum value
Log Registry Value Error	0	0	1

12.7.6 Defining the Operator Communications Interval Setting

If an I/O error occurs, the OpenVMS Registry server can display a message to the operator console using OPCOM. The **Operator Communications Interval** setting determines how long the OpenVMS Registry server waits after the I/O error to determine if the error is going to persist. If the error does persist, OpenVMS Registry writes a message to the operator console.

By default, the OpenVMS Registry server writes a message to the operator console if the error persists longer than 60 seconds.

Setting Name	Default value	Minimum value	Maximum value
Operator Communication Interval	60	30	120

12.7.7 Defining the Process Time Limit Setting

The OpenVMS Registry server writes a message to the server log file if it takes too long to process a request. The **Process Time Limit** setting determines when a request has taken too long.

By default, 180 seconds are allowed per request before the OpenVMS Registry logs a message.

Setting Name	Default value	Minimum value	Maximum value
Process Time Limit	180	60	600

12.7.8 Defining the Reply Log Cleaner Interval Setting

The OpenVMS Registry server maintains a log of recent replies that it uses to reconstruct work in progress in the case of failover. After a specified time, the server discards these replies. The **Reply Log Cleaner Interval** setting determines how often the OpenVMS Registry discards these replies.

By default, the OpenVMS Registry server discards replies every five seconds.

Setting Name	Default value	Minimum value	Maximum value
Reply Log Cleaner Interval	5	5	60

12.7.9 Defining Snapshot Interval/Snapshot Location/Snapshot Versions Settings

The OpenVMS Registry server maintains backup copies of the OpenVMS Registry database. The **Snapshot Interval** setting determines how often the OpenVMS Registry server creates a backup copy. The **Snapshot Location** setting determines where the OpenVMS Registry stores the copy. The **Snapshot Versions** setting determines how many previous copies the OpenVMS Registry keeps.

By default, the OpenVMS Registry database is copied to backup once per day. By default, the OpenVMS Registry database is copied to the location determined by the definition of the `SYS$REGISTRY` logical name. By default, the OpenVMS Registry keeps five previous versions of the OpenVMS Registry database.

Setting Name	Default value	Minimum value	Maximum value
Snapshot Interval	86400	3600	604800
Snapshot Location	`SYS$REGISTRY`	—	—
Snapshot Versions	5	1	10

12.7.10 Defining the Write Retry Interval Setting

If the OpenVMS Registry finds an error when writing to the OpenVMS Registry database, the OpenVMS Registry server retries the write at an interval specified by the **Write Retry Interval** setting.

By default, the OpenVMS Registry server attempts to retry failed writes to the OpenVMS Registry database every five seconds.

Setting Name	Default value	Minimum value	Maximum value
Writer Retry Interval	5	1	30

13

OpenVMS Registry System Management

13.1 Installing the OpenVMS Registry

The OpenVMS Registry server is installed as part of the OpenVMS system installation.

Before you can use the OpenVMS Registry, you must configure the OpenVMS Registry server.

The first time you start the OpenVMS Registry server using the startup process described in Section 13.2, the OpenVMS system creates the OpenVMS Registry database.

You can access the OpenVMS Registry in several ways. Depending on how you want to access the OpenVMS Registry, you must install the following products:

- If you want to access the OpenVMS Registry using the COM APIs, you must install COM for OpenVMS and populate the OpenVMS Registry database. For more information about installing COM for OpenVMS, see Chapter 4. For more information about populating the OpenVMS Registry database, see Section 6.2.
- If you want to access the OpenVMS Registry using the Windows application `RegEdt32`, you must first install, configure, and start HP Advanced Server for OpenVMS. For more information, see the HP Advanced Server for OpenVMS documentation.

You can also access the OpenVMS Registry using the OpenVMS Registry server management utility or the OpenVMS Registry system services, which are installed as part of the OpenVMS Registry.

The OpenVMS Registry configuration procedure (REG$CONFIG) provides information about the OpenVMS Registry server status and the OpenVMS Registry database location, and allows you to change OpenVMS Registry logical names and paths.

Enter the following command to invoke the OpenVMS Registry configuration procedure:

```
$ @SYS$MANAGER:REG$CONFIG
```

The system displays the following menu:

```
------------------------------------------------------
        OpenVMS Registry Configuration Utility
        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        1 - Configure OpenVMS Registry logical names and directory paths
        2 - Display OpenVMS Registry logical names and directory paths
        3 - Check the state of the OpenVMS Registry server
        4 - Start the OpenVMS Registry server on this node
        5 - Convert to latest database version and/or reclaim database
        H - Help about this utility
       [E] - Exit
Please enter your choice :
------------------------------------------------------
```

To select an option, enter the option number. The options are as follows:

- `1 - Configure OpenVMS Registry logical names and directory paths`

  Allows you to configure the OpenVMS Registry server startup value and specify the location of the OpenVMS Registry database.

  For this procedure, see Section 13.1.1.

- `2 - Display OpenVMS Registry logical names and directory paths`

  Displays the current values of the OpenVMS Registry server logical (startup value) for this node and the OpenVMS Registry database location.

- `3 - Check the state of the OpenVMS Registry server`

  Displays the current state of the OpenVMS Registry server. The system displays one of the following:

  ```
  The OpenVMS Registry server is started in the cluster.
  The OpenVMS Registry server is started on this node.
  The OpenVMS Registry server is not started.
  ```

- `4 - Start the OpenVMS Registry server on this node`

  Starts the OpenVMS Registry server on the current node. The system displays the following message:

  ```
  The OpenVMS Registry server has successfully started.
  ```

- `5 - Convert to latest database version and/or reclaim database`

  This option can be used to convert an existing database to the latest version. The latest version is determined by the currently running server.

  This option can also be used to compact the database. Over time, the database can become fragmented in much the same way that a disk can become fragmented. Compacting the database makes it more efficient and requires less disk space.

- `H - Help about this utility`

  Displays online help for OpenVMS Registry Configuration utility options.

- `[E] - Exit`

  Exits the OpenVMS Registry Configuration utility.

Tip: Enter Q (Quit) at any time

You can enter Q at any prompt to return to the OpenVMS Registry Configuration utility menu.

If you quit while you are configuring logical names, the system updates only those values for which you have received a confirmation message.

13.1.1 Configuring OpenVMS Registry Values

The system displays the following questions:

1. The system prompts you to enter standalone or cluster information. The system displays the following message:

```
Is this system now a node in a cluster or will this system
become part of a cluster? (Y/N/Q):
```

2. The system displays the current information about the REG$TO_BE_STARTED logical, then prompts you to change the value.

```
            - REG$TO_BE_STARTED -

   [current value of REG$TO_BE_STARTED]

        NOTE: Setting this logical to TRUE starts the OpenVMS Registry
          server automatically when the system boots. Setting this logical
          to FALSE prevents the OpenVMS Registry server from starting
          when the system boots and prevents other products from starting
          the OpenVMS Registry server. If the OpenVMS Registry Server is not
          started at boot time, but other products that require an OpenVMS
          Registry server are able to start the OpenVMS Registry server, you
          do not need to assign a value to this logical.

Do you want to change this value? (Y/N/Q) [Y]:
```

If you choose Y, the system prompts you for the new value.

```
Enter the new value (TRUE/FALSE/NOVAL/Q):
```

Enter one of the following:

Action	Value
Start the OpenVMS Registry server on reboot. Allow other products to start the server.	TRUE
Do not start the OpenVMS Registry server on reboot. Do not allow other products to start the server.	FALSE
Do not start the OpenVMS Registry server on reboot. Allow other products to start the server. (Deassigns the logical name.)	NOVAL
Quit this procedure and return to the OpenVMS Registry Configuration utility menu.	Q

```
In which logical name table do you want the logical defined?
       (SYSTEM/SYSCLUSTER/Q) :
```

Enter one of the following:

Action	Value
Add the REG$TO_BE_STARTED logical to the LNM$SYSTEM logical name table. This table contains names that are shared by all processes in the system.	SYSTEM
Add the REG$TO_BE_STARTED logical to the LNM$SYSCLUSTER logical name table. This table contains names that are shared by all processes in an OpenVMS Cluster.	SYSCLUSTER
Quit this procedure and return to the OpenVMS Registry Configuration utility menu.	Q

After you enter the new or updated value, the system confirms the change and displays the line you must add to your SYLOGICALS.COM file, if you have selected the SYSTEM table, or to the end of the SYSTARTUP_VMS.COM file, if you have selected the SYSCLUSTER table.

```
        The logical REG$TO_BE_STARTED has been temporarily defined.
        Before you reboot the system you must edit your SYLOGICALS.COM
        to include the line:

        DEFINE/TABLE=table-name REG$TO_BE_STARTED value

Press [Enter] to continue.
```

3. The system displays the current information about the SYS$REGISTRY logical, then prompts you to change the value.

```
             - SYS$REGISTRY logical -

   current value of SYS$REGISTRY

        Note: When the OpenVMS Registry server is started, the system
          creates an OpenVMS Registry database at this location.
          If an OpenVMS Registry database already exists on your system,
          you must redefine the SYS$REGISTRY logical to point to the
          existing OpenVMS Registry database location.

Do you wish to change this value? (Y/N/Q) [Y]:
```

If you choose Y, the system prompts you for the new value.

```
Enter the new value for SYS$REGISTRY ("yourvalue"/NOVAL/Q):
```

Enter one of the following:

Action	Value
Define a new or changed location for the OpenVMS Registry database.	A valid directory specification, such as DKA0:[SYS$REGISTRY].
Deassign the logical name.	NOVAL
Quit this procedure and return to the OpenVMS Registry Configuration utility menu.	Q

4. The system displays your updated value and prompts you to confirm the value.

```
 You have entered:  value
 Is this correct? (Y/N/Q) [Y]:
```

5. The system prompts you to enter a logical table name in which to store the new or updated logical.

```
In which logical name table do you want the logical defined?
       (SYSTEM/SYSCLUSTER/Q):
```

Enter one of the following:

Action	Value
Add the SYS$REGISTRY logical to the LNM$SYSTEM logical name table. This table contains names that are shared by all processes in the system.	SYSTEM
Add the SYS$REGISTRY logical to the LNM$SYSCLUSTER logical name table. This table contains names that are shared by all processes in an OpenVMS Cluster.	SYSCLUSTER
Quit this procedure and return to the OpenVMS Registry Configuration utility menu.	Q

After you enter the new or updated value, the system confirms the change and displays the line you must add to your SYLOGICALS.COM file, if you have selected the SYSTEM table, or to the end of the SYSTARTUP_VMS.COM file, if you have selected the SYSCLUSTER table.

```
       The logical SYS$REGISTRY has been temporarily defined.
       Before you reboot the system you must edit your SYLOGICALS.COM file
       to include the line:

       DEFINE/TABLE=table-name SYS$REGISTRY dir-spec

Press [Enter] to continue.
```

6. The system displays information about the location of the OpenVMS Registry database.

```
            - SYS$REGISTRY directory -

  [directory status]
```

If the directory does not exist, the system prompts you to create the directory.

```
       !!Caution!!  When the OpenVMS Registry server starts, the system
         creates an OpenVMS Registry database at this location. If you
         already have an OpenVMS Registry database on your system, you must
         redefine the SYS$REGISTRY logical to point to that location.

Do you wish to create the directory? (Y/N/Q) [Y]:
```

If you enter Y the system confirms the directory creation.

```
       The SYS$REGISTRY directory has now been created.

Press [Enter] to return to the menu.
```

13.1.2 Registry Database Conversion and Compaction

Beginning with OpenVMS Version 7.3-1, the OpenVMS Registry supports two database formats, Version 1 and Version 2. The Version 2 database format includes an index access that improves on the access performance provided in the Version 1 database format.

The Registry server supports both database formats. To take advantage of the indexing feature, you must convert a Version 1 database to the Version 2 format.

Who Should Convert

Converting the Registry database is optional. Large databases are more likely to benefit from converting to Version 2. The more subkeys or values on a key, the more likely it is that there will be a performance improvement from using the Version 2 database. For databases where there are only one or two subkeys or values on most keys, converting to Version 2 will yield minimal performance improvement.

Who Should Not Convert

Do not convert the Registry database to Version 2 if you plan to run Registry servers on nodes in a mixed-version cluster. Registry servers on nodes running versions of OpenVMS prior to Version 7.3-1 cannot access a Version 2 database. Operating in this manner is not supported.

13.1.2.1 Converting an Existing Database

To convert an existing database to Version 2, invoke REG$CONFIG and select step 5, "Convert to latest database version and/or reclaim database," and follow the instructions. Be sure to invoke REG$CONFIG on a node that normally runs the Registry server. You cannot run REG$CONFIG on a node that explicitly disables the Registry server, that is, a node on which the REG$TO_BE_STARTED logical is FALSE.

The command procedure saves a copy of the current database in a separate directory, both in binary and EXPORT command (ASCII text) format.

The procedure requires an interruption in Registry services, so execute it at a time when this will cause minimal disruption. Prior to shutting down the Registry server, you must also shut down all layered products that use the Registry: COM for OpenVMS, Advanced Server for OpenVMS, and any other third-party applications.

Be sure to shut down these applications on all nodes in the cluster.

The command procedure informs you when to shut down these services and pauses while you perform this task. You should shut down in this order:

1. Shut down any third-party applications that use Registry.
2. Shut down COM for OpenVMS using SYS$STARTUP:DCOM$SHUTDOWN.COM.
3. Shut down Advanced Server for OpenVMS using SYS$STARTUP:PWRK$SHUTDOWN.COM.
4. Shut down the Registry server on all nodes in the cluster.

13.1.2.2 Determining the Registry Database Version

There are several ways to determine the version of a given Registry database.

Note

In all of the following cases, the database version is a longword (DWORD). The lower word is the major version, and the upper word is the minor version. Normally the minor version is zero.

- For a database currently in use, enter the command:

```
$ REG$CP == "$REG$CP"
$ REG$CP LIST VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY
```

  The value *Database Version* contains the version of the database currently loaded. Database versions prior to Version 2 do not have a Database Version value.

- For an exported database, look for the following statement in the exported database file:

```
Database Version"=dword:00000002"
```

  Exported databases prior to Version 2 do not contain this statement.

- The files SYS$MANAGER:REGISTRY$SERVER.LOG and .ERR identify the version of the database opened by the Registry server when it starts up.

- For a given set of registry files, dump the REGISTRY$ROOT.DAT file. The second longword indicates the database version.

13.1.2.3 Reclaiming the Database

You can also use Step 5 in REG$CONFIG to reclaim the database, which reclaims wasted space that can occur over time as the database becomes fragmented. Follow the same instructions as for converting the database. You can repeat this procedure as often as you like.

REG$CONFIG does not allow you to reclaim a Version 1 database. However, you can manually reclaim a Version 1 database by performing the steps in the following section.

13.1.2.4 Manual Conversion and Reclamation

Conversion and reclamation of a database use the Registry Import/Export feature. The current Registry database is exported, a new database is created, then the original database is imported. You can perform these steps manually as follows:

1. Shut down all applications in the cluster that use Registry (described in Section 13.1.2.1), including COM for OpenVMS and Advanced Server.

2. Enter the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP EXPORT DATABASE/OUTPUT=(filespec)
```

   If you do not specify a value for /OUTPUT, it defaults to REGISTRY.TXT in your current default directory. The output format defaults to /FORMAT=VMS. Do not specify /FORMAT=NT in this step.

3. Shut down the Registry server on all nodes in the cluster using the following command:

```
$ SET SERVER REGISTRY/CLUSTER/EXIT
```

4. Preserve the current database by copying all of the files in SYS$REGISTRY to a separate directory.

5. Delete all the files in SYS$REGISTRY with the exception of REGISTRY$CONFIGDONE.DAT.

6. Start the Registry server on one node in the cluster using the following command:

```
$ SET SERVER REGISTRY/START [/NODE=node]
```

7. Create a new Registry database using the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE DATABASE
```

Note

The new database will default to a Version 2 database. To reclaim without converting to Version 2, enter the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE DATABASE/VERSION=1
```

8. Import the original database using the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP IMPORT/INPUT=(filespec-from-step-2)
```

If you do not specify a value for /INPUT, it defaults to REGISTRY.TXT in your current default directory.

9. Start the Registry server on any remaining nodes on which you want it to run (see Step 6).
10. Restart any applications that were stopped in Step 1.

13.2 Starting the OpenVMS Registry

You can control how the OpenVMS Registry will start as follows:

- Start the OpenVMS Registry automatically when the system reboots.
- Have products that require the OpenVMS Registry to be running start the OpenVMS Registry.
- Start the OpenVMS Registry manually.
- Prevent the OpenVMS Registry from starting.

Use the OpenVMS Registry Configuration utility to control how the OpenVMS Registry starts.

13.2.1 Starting the OpenVMS Registry Manually

Under some conditions, you might want to start the OpenVMS Registry server manually.

HP recommends that you use the SYS$STARTUP:REG$STARTUP.COM command procedure. The following command procedure ensures that the server process quotas are set to the required minimum values:

```
$ @SYS$STARTUP:REG$STARTUP.COM
```

Alternately, you can use the following command to start the OpenVMS Registry manually:

```
$ SET SERVER REGISTRY_SERVER/START
```

13.3 Shutting Down the OpenVMS Registry

The OpenVMS Registry server is shut down automatically as part of a system shutdown.

If you want to shut down the OpenVMS Registry manually, use the following command:

```
$ SET SERVER REGISTRY_SERVER/EXIT
```

13.4 OpenVMS Registry Server Commands

The OpenVMS Registry server commands allow you to display (SHOW) and change (SET) the state of the OpenVMS Registry server. The following sections list and describe the OpenVMS Registry server commands.

SHOW SERVER REGISTRY_SERVER

Show the current status of the OpenVMS Registry on a specified node.

This command requires the SYSPRV privilege.

Format

SHOW SERVER REGISTRY_SERVER

[/MASTER | /CLUSTER | /NODE=(*node*,...)]

[/PAGE]

Qualifiers

/MASTER
Displays the node and process ID (PID) of the current OpenVMS Registry master server in the cluster. This command does not communicate with the OpenVMS Registry servers in the cluster. Requires the SYSPRV privilege.

/CLUSTER
Returns the show output from each OpenVMS Registry server in the cluster, listing the OpenVMS Registry master server information first.

/NODE=(*node*,...)]
Returns OpenVMS Registry server information about the servers on the specified nodes, listed in the order in which you enter the node names. The node names you specify must be in the current cluster.

/PAGE
Displays the returned show output in a scrollable page display.

SET SERVER REGISTRY_SERVER

Change the state of the OpenVMS Registry.

This command requires the SYSPRV privilege.

Format

SET SERVER REGISTRY_SERVER

[/MASTER | /CLUSTER | /NODE=(*node*,...)]

[/START | /RESTART | /EXIT | /ABORT]

[/[NO]LOG]

Qualifiers

/MASTER
Issues the specified command to the OpenVMS Registry master server only. Requires the SYSPRV privilege.

/CLUSTER
Issues the SET command to each OpenVMS Registry server in the cluster, setting the OpenVMS Registry master server last.

/NODE=(*node*,...)
Issues the SET command to the OpenVMS Registry servers on the specified nodes, in the order in which you enter the node names. The node names must be in the current cluster.

/START[=(*node*,...)]
Starts the OpenVMS Registry server on the specified node or nodes in the cluster.

/EXIT[=(*node*,...)]
Stops the OpenVMS Registry server on the specified node or nodes in the cluster.

/ABORT[=(*node*,...)]
Aborts the OpenVMS Registry server on the specified node or nodes in the cluster.

/[NO]LOG
Creates a new OpenVMS Registry log file in SYS$REGISTRY. NOLOG is the default.

13.5 OpenVMS Registry Failover in a Cluster

To increase the availability and reliability of the OpenVMS Registry, you can run multiple OpenVMS Registry servers in a cluster, up to one per node. No matter how many OpenVMS Registry servers you run, you have only one OpenVMS Registry database.

There can be more than one Registry server in a cluster, but only one server functions as the master server. The other servers function as backup servers in case the master server or the node it is running on fails.

By default, the first OpenVMS Registry server process that is active in the cluster remains active until either the process no longer exists or the priority among OpenVMS Registry server processes changes.

13.5.1 Changing the Priority of OpenVMS Registry Server Processes

You can change the priority of OpenVMS Registry server processes by creating and modifying the priority value of each node in the cluster that will run the OpenVMS Registry server process: the higher the value, the higher the priority.

The Registry servers use a priority scheme to determine which server is the master. This scheme is provided so that a system manager can weigh one server against another, depending on the system capabilities. For example, the system manager may want the server master to normally run on the fastest system. This priority is determined by a Registry value and should not be confused with process priority.

Setting the priority of a Registry server is not required. If the server priorities have not been set, all servers run at the same priority. In this case, the server that starts executing first is the master server.

Example 13–1 shows priority values being assigned so that NODENAME1 will be the active OpenVMS Registry server process in the cluster.

Example 13–1 Setting Priority Values

```
$ REG$CP == "$REG$CP"
$ CREATE VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY\PRIORITY -
_$ /NAME=NODENAME1/DATA=15/TYPE=DWORD
$ REG$CP CREATE VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY\PRIORITY -
_$ /NAME=NODENAME2/DATA=10/TYPE=DWORD
$ REG$CP CREATE VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY\PRIORITY -
_$ /NAME=NODENAME3/DATA=5/TYPE=DWORD
```

In Example 13–1, if NODENAME1 shuts down, control of the OpenVMS Registry database passes to the server process on NODENAME2.

Example 13–2 shows the system manager increasing the priority value of NODENAME3 to 20.

Example 13–2 Changing Priority Values

```
$ REG$CP == "$REG$CP"
$ REG$CP MODIFY VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY\PRIORITY -
_$ /NAME=NODENAME3/DATA=20/TYPE=DWORD
```

In Example 13–2, the OpenVMS Registry server process on NODENAME1 goes into standby mode and the OpenVMS Registry server process on NODENAME3 becomes active.

13.6 Connecting to the OpenVMS Registry from a Windows System

To connect to the OpenVMS Registry from a Windows system, you must do the following:

- On the OpenVMS system:
  - — Install the HP Advanced Server for OpenVMS.
  - — Configure the HP Advanced Server for OpenVMS.
- On the Windows system:
  - — Install and configure any required hardware.
  - — Install and configure the Windows Server or Workstation software.

When you access the OpenVMS Registry database from a Windows system, you will have all the privileges granted on your Windows system. For example, if you are logged on to the Windows system as an Administrator, you will be able to read and write to all keys and values in the OpenVMS Registry. Access to OpenVMS Registry keys is based on your Windows user profile (username and Group membership). Connect to the OpenVMS Registry through HP Advanced Server for OpenVMS; use the Windows Regedt32 application to view and change keys, values, and security settings.

Caution

Be careful when you modify OpenVMS Registry database keys and values. If you damage the OpenVMS Registry database, you can affect applications that use the Registry on the entire OpenVMS system or cluster.

13.7 OpenVMS Registry Quotas

A quota mechanism limits the size of the OpenVMS Registry database. The system assigns a quota to the root key datafile for every OpenVMS Registry file. By default, these root keys are the USERS key (REGISTRY$USERS.REG) and the LOCAL_MACHINE key (REGISTRY$LOCAL_MACHINE.REG).

The quota limits the size of the information contained within the file but does not include the size of information stored in other files, even if the files are part of the subtree.

The default quota and file-specific quotas are stored in the OpenVMS Registry under the HKEY_LOCAL_MACHINE\SYSTEM\Registry key. For more information about these keys, see Section 12.3.

13.8 OpenVMS Registry Security

A user can access (read and modify) the OpenVMS Registry directly in the following ways:

- From a Windows system (through a connection through HP Advanced Server for OpenVMS)
- Using the OpenVMS Registry system services ($REGISTRY[W])
- Using the OpenVMS Registry server management utility (`REG$CP`)

For a discussion of what system privileges and right identifiers each user needs, see Section 12.6.1. For a description of how to grant the necessary system privileges and right identifiers, see Section 12.6.1.1.

You can change a key's security attributes only from a Windows system—you cannot change a key's security attributes from an OpenVMS system. OpenVMS does not create or manage Windows security attributes.

13.9 Backing Up and Restoring the OpenVMS Registry Database

The OpenVMS Registry includes a server management utility that allows you to back up and restore the entire OpenVMS Registry database to or from a file from the OpenVMS DCL prompt as long as you have the required system privileges.

For more information about backing up and restoring the OpenVMS Registry database, see Section 14.2 and the `REG$CP` server management utility CREATE SNAPSHOT command and the EXPORT command.

13.10 Internationalization and Unicode Support

To integrate with Windows, the OpenVMS Registry is Unicode compliant. For more information about Unicode, see the *OpenVMS Guide to Extended File Specifications*.

14

OpenVMS Registry Server Management

14.1 Managing the OpenVMS Registry Server from the Command Line

The OpenVMS Registry includes a server management utility that allows you to update and display OpenVMS Registry information from the OpenVMS DCL prompt.

The utility also allows you to back up and restore the entire OpenVMS Registry database to or from a file, as long as you have the required system privileges. For more information about backing up and restoring the OpenVMS Registry database, see Section 14.2 and the CREATE SNAPSHOT, EXPORT, and IMPORT commands in the command reference section of this chapter.

To start the OpenVMS Registry server management utility, enter one of the following commands:

```
$ RUN SYS$SYSTEM:REG$CP
```

Note

Before you can access the OpenVMS Registry database, the OpenVMS Registry server must be running either in the cluster or on the standalone system.

Table 14–1 lists and describes OpenVMS Registry server management utility commands.

Table 14–1 OpenVMS Registry Server Management Utility Commands

Command	Identifier	Action
CREATE DATABASE	SYSPRV	Creates a new set of OpenVMS Registry database files.
CREATE KEY	REG$UPDATE	Creates one or more keys in the OpenVMS Registry database.
CREATE SNAPSHOT	SYSPRV	Makes an immediate backup of the OpenVMS Registry database files.
CREATE VALUE	REG$UPDATE	Specifies the data component for a key.
DELETE KEY	REG$UPDATE	Removes one or more keys from the OpenVMS Registry database.

(continued on next page)

Table 14–1 (Cont.) OpenVMS Registry Server Management Utility Commands

Command	Identifier	Action
DELETE TREE	REG$UPDATE	Removes the specified key and all of its subkeys from the OpenVMS Registry database.
DELETE VALUE	REG$UPDATE	Removes one or more values from a specified key.
EXIT		Enables you to exit from REG$CP and return to the DCL command prompt.
EXPORT DATABASE	SYSPRV	Exports the entire OpenVMS Registry database file to a text format file.
EXPORT KEY	SYSPRV	Exports a specific OpenVMS Registry key (and optionally subkeys) to a text format file.
HELP		Displays information about using the REG$CP utility and includes formats and explanations for commands, parameters, and qualifiers.
IMPORT	REG$UPDATE	Imports a text format version of a registry database to the OpenVMS Registry format.
LIST KEY	REG$LOOKUP	Displays all subkey information for a specified key.
LIST SECURITY DESCRIPTOR	REG$LOOKUP	Displays the security descriptor associated with the specified key.
LIST VALUE	REG$LOOKUP	Displays all values of a specified key.
MODIFY KEY	REG$UPDATE	Modifies the information of a specified key.
MODIFY TREE	REG$UPDATE	Modifies the information of a specified key and its subkeys.
MODIFY VALUE	REG$UPDATE	Modifies the information of a specified value.
SEARCH KEY	REG$LOOKUP	Displays the path name of all keys that match a specified key.
SEARCH VALUE	REG$LOOKUP	Displays the path name of all keys that match a specified value name.
SHOW COUNTERS	REG$PERFORMANCE	Displays counter information.
SHOW FILE	REG$PERFORMANCE	Displays OpenVMS Registry database file statistics.
SPAWN		Creates a subprocess of the current process.
START MONITORING	REG$PERFORMANCE	Enables monitoring functions.
STOP MONITORING	REG$PERFORMANCE	Disables monitoring functions.
WAIT		Waits for the specified number of seconds, or waits for a random number of seconds within a range of seconds for command completion.

(continued on next page)

Table 14–1 (Cont.) OpenVMS Registry Server Management Utility Commands

Command	Identifier	Action
ZERO COUNTERS	REG$PERFORMANCE	Resets monitoring counters.

Note

A user who has the SYSPRV privilege can can execute all the commands listed in Table 14–1. You must specify an OpenVMS Registry identifier only if the user does not have SYSPRV privilege.

If you grant a user the REG$UPDATE identifier, in addition to the commands listed in Table 14–1, the user can also execute the following commands:

```
LIST KEY
LIST VALUE
SEARCH KEY
SEARCH VALUE
```

If you do not grant a user the REG$LOOKUP identifier, the user cannot examine keys (LIST KEY, LIST VALUE, SEARCH KEY, SEARCH VALUE, or EXPORT KEY or DATABASE) that are protected from world read access. However, the default for almost all keys created in a new database is world read access. The exceptions are keys that have ACLs that prohibit world read access. For example, a user who does not have the REG$LOOKUP identifier cannot execute the following command:

```
$ REG$CP == "$REG$CP"
$ REG$CP LIST KEY HKEY_LOCAL_MACHINE\SECURITY
```

14.2 Backing Up and Restoring the OpenVMS Registry Database

The REG$CP server management utility includes two commands that allow you to back up and restore an OpenVMS Registry database.

- The EXPORT command allows you to back up the OpenVMS Registry keys and values on demand in OpenVMS or Windows format.

  You can use this command to export part or all of an OpenVMS Registry database. The corresponding IMPORT command allows you to restore or import OpenVMS Registry or Windows Registry keys and values.

  Note

  The EXPORT command defaults to /FORMAT=VMS. You should always use VMS format when preserving the OpenVMS Registry, because the backup copy could be used as input to the IMPORT command on an OpenVMS system. The NT Export Registry File format does not preserve OpenVMS specific data structures.

  For more information, see the EXPORT and IMPORT commands in the command reference section of this chapter.

- The CREATE SNAPSHOT command allows you to back up the OpenVMS Registry database files automatically on a specified schedule.

By default, the REGISTRY_SERVER process creates a snapshot of the OpenVMS Registry database every 24 hours. You can change this interval by modifying the **Snapshot Interval** setting in the OpenVMS Registry server operations. (For more information about these operations, see Section 12.7.)

The following example shows how to modify the interval between automatic snapshots of the OpenVMS Registry database from the default of once every 24 hours to once every hour.

```
$ REG$CP == "$REG$CP"
$ REG$CP MODIFY VALUE HKEY_LOCAL_MACHINE\SYSTEM\REGISTRY -
_$ /NAME="Snapshot Interval"/DATA=3600/TYPE=DWORD
```

For more information, see the CREATE SNAPSHOT command in this chapter.

14.2.1 Creating a Snapshot of the OpenVMS Registry Database

Use the following procedure to create a snapshot of the OpenVMS Registry database:

1. Verify that the REGISTRY_SERVER process is running in the cluster.
2. From an account with the SYSPRV privilege, enter the following commands:

```
$ REG$CP == "$REG$CP"
$ REG$CP CREATE SNAPSHOT
```

The resulting snapshot consists of the following two files, located in the specified directory:

```
REGISTRY$LOCAL_MACHINE.RSS
REGISTRY$USERS.RSS
```

14.2.2 Restoring a Snapshot of the OpenVMS Registry Database

Note

Before you execute the following procedure, verify the location of the OpenVMS Registry snapshot files REGISTRY$LOCAL_MACHINE.RSS and REGISTRY$USERS.RSS. By default, these files are placed in the SYS$REGISTRY directory. However, they may have been moved or written to a different location. The server writes the files to the location specified in the Snapshot Location value on the HKEY_LOCAL_MACHINE\SYSTEM\Registry key.

Use the following procedure to restore a snapshot of the OpenVMS Registry database:

1. Shut down all layered products and applications that use the Registry server on all nodes in the cluster.
2. Shut down the OpenVMS Registry server on all nodes in the cluster. (For information about shutting down the OpenVMS Registry, see Section 13.3.)
3. Preserve all files in SYS$REGISTRY in a separate directory.
4. Do one of the following:
   - Delete all files in SYS$REGISTRY, and then invoke SYS$STARTUP:REG$CONFIG.COM to configure and start the OpenVMS

Registry server. By starting the Registry server in this step, you are creating a new database and a new REGISTRY$CONFIGDONE.DAT file.

or:

- Delete all files in SYS$REGISTRY except REGISTRY$CONFIGDONE.DAT and start the OpenVMS Registry server by invoking SYS$STARTUP:REG$STARTUP.COM.

5. Shut down the OpenVMS Registry server.
6. Delete the files SYS$REGISTRY:REGISTRY$LOCAL_MACHINE.REG and SYS$REGISTRY:REGISTRY$USERS.REG.
7. Copy the OpenVMS Registry snapshot files to SYS$REGISTRY and rename them as follows:

```
$ RENAME REGISTRY$LOCAL_MACHINE.RSS REGISTRY$LOCAL_MACHINE.REG
$ RENAME REGISTRY$USERS.RSS REGISTRY$USERS.REG
```

8. Start the OpenVMS Registry server. (See Section 13.2.1.)

14.3 OpenVMS Registry Server Management Utility Syntax

The following command section describes each OpenVMS Registry command in alphabetical order.

Note

In all the commands in this section, the **key-name** parameter is a string that specifies the full path of the key, beginning from one of following entry points:

```
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CLASSES_ROOT
```

You can also specify the strings `REG$_HKEY_LOCAL_MACHINE`, `REG$_HKEY_USERS`, and `REG$_HKEY_CLASSES_ROOT`.

For all server management commands, links are not followed. (For more information about links, see Section 12.2.1.3.)

To make key and values names case sensitive, enclose the keys and values in quotation marks (for example: `"value"`).

CREATE DATABASE

Creates the basic OpenVMS Registry database files in the location specified by the SYS$REGISTRY logical. The command creates an empty database and loads the predefined keys.

If you enter this command and the database files already exist, the utility does not overwrite the existing files. The system displays a warning that the files already exist. If you want to create a new OpenVMS Registry database, you must first delete all previous versions of the database files. If you delete the OpenVMS Registry database files, you will lose all keys, subkeys, and values stored in the OpenVMS Registry.

This command requires the SYSPRV privilege.

The following table lists and describes the OpenVMS Registry database files.

File	Description
REGISTRY$ROOT.DAT	Root of the database
REGISTRY$USERS.REG	HKEY_USERS tree
REGISTRY$LOCAL_MACHINE.REG	HKEY_LOCAL_MACHINE tree
REGISTRY$MASTER.RLG	The master commit log file
REGISTRY$REPLY.RLG	Log file that tracks modification requests to the OpenVMS Registry database

Format

CREATE DATABASE

Parameters

None

Qualifiers

/VERSION=*version-number*
Specifies how to format the database. Specify a version number of 1 to create a non-indexed database. Specify a version number of 2 to create an indexed database. If unspecified, the default value is 2.

Note

OpenVMS Registry servers running on OpenVMS V7.3-1 or later support both version 1 and version 2 databases. Registry servers prior to OpenVMS V7.3-1 support version 1 databases only.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default

interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> CREATE DATABASE
```

By default, this command regenerates the basic OpenVMS Registry database files using version 2 format (indexed database).

```
REG> CREATE DATABASE/VERSION=1
```

This command causes the basic OpenVMS Registry database files to be regenerated using version 1 format (non-indexed database).

CREATE KEY

Creates one or more keys in the OpenVMS Registry database.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

CREATE KEY key-name [,...]

Parameters

key-name[,...]
Specifies the name of the key to create. You can create multiple keys by separating the keys with commas.

Qualifiers

/CACHE_ACTION=*value*
Specifies the cache attribute for the new key. The *value* can be WRITEBEHIND (write to disk later) or WRITETHRU (write to disk immediately).

If you omit /CACHE_ACTION, the system creates the key with the cache attribute set to REG$K_WRITEBEHIND.

/CLASS_NAME=*string*
Specifies the class name of the key.

/LINK=(TYPE=*value*, NAME=*key-name*)
Defines the key as a link to another key. The link value must be one of the following:

- SYMBOLICLINK
- NONE

To remove a link, enter the following:

```
/LINK=(TYPE=NONE,NAME="")
```

/SECPOLICY=*policy*
Defines the security policy for the key. Currently the only valid policy is NT_40.

/VOLATILE=*level*
/NONVOLATILE (default)
Specifies whether or not the new key is volatile. If you are running the OpenVMS Registry on a standalone OpenVMS system, volatile keys are lost when the system reboots. If you are running the OpenVMS Registry in an OpenVMS cluster, volatile keys are lost when all nodes in the cluster are rebooted.

The values for *level* are as follows:

- NONE (same as /NONVOLATILE)
- CLUSTER

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> CREATE KEY/CACHE_ACTION=WRITEBEHIND
HKEY_USERS\GUEST, HKEY_USERS\SYSTEM
```

Creates the GUEST and SYSTEM keys under the HKEY_USERS entry point. The keys are created with the write-behind attribute.

CREATE SNAPSHOT

Creates a snapshot of the OpenVMS Registry database. That is, the system writes all cached OpenVMS Registry keys or values and makes a copy of the OpenVMS Registry database files.

The OpenVMS Registry server copies database files to the location specified by /DESTINATION (SYS$REGISTRY by default), using the file extension .RSS (Registry SnapShot). To restore the snapshot, shut down all applications using the Registry, shut down the Registry server, and copy the files to SYS$REGISTRY, renaming them with the .REG file extension.

Note

When you restore the database from a snapshot, you lose all modifications that were made to the database since the last snapshot was taken.

By default the OpenVMS Registry server creates a snapshot automatically every 24 hours and retains the five most recent snapshot files.

This command requires the SYSPRV privilege.

Format

CREATE SNAPSHOT

Parameters

None

Qualifiers

/DESTINATION=*file-spec*
Controls where the system will write the snapshot files. By default, the system creates the snapshot in the location specified by the `SYS$REGISTRY` logical.

If you specify the /DESTINATION qualifier but do not provide a valid directory, the system creates the snapshot files in the directory in which you started the OpenVMS Registry server.

/VERSIONS=*number*
Specifies how many previous versions of the snapshot files to keep.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> CREATE SNAPSHOT/DESTINATION=SYS$REGISTRY/VERSION=3
```

Creates a snapshot of the OpenVMS Registry database in the SYS$REGISTRY directory. If more than three versions of the OpenVMS Registry database snapshot files exist, the system deletes the oldest version (the same as purge/keep=3 command).

CREATE VALUE

Specifies the data component for the specified key. If the value does not exist, the command creates the value.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

CREATE VALUE key-name

Parameters

key-name
Specifies the name of the key for which you will set the value.

Qualifiers

/DATA=*value*
The value can be one of the following:

- A string (for example, /DATA=COSMOS)
- An array of strings separated by a comma and enclosed in parentheses (for example, /DATA=(COSMOS,Noidea)
- A longword in binary, octal, decimal, or hexadecimal format. %B, %O, %D, and %X, or 0B, 0O, 0D, and 0X prefixes specify the format. The default is decimal.

  Examples:

```
/DATA=%X1A0FCB (hex)
/DATA=0X1A0FCB (hex)
/DATA=D1234 (decimal)
/DATA=3D1234 (decimal, by default)
```

Note

You cannot specify the /INPUT qualifier with /DATA.

/INPUT=*filename*
Specifies that the value data is to be read from a file.

The input value data can be specified in one of the following formats. With the exception of SZ, the format is specified by a keyword at the start of the file. The keyword can be entered in uppercase or lowercase. Input records following a keyword can span multiple lines; use "\" at the end of any continuation lines.

- SZ: a null-terminated Unicode string

  Enter SZ data by enclosing the record in quotes. You do not specify an SZ keyword for this input value type. You cannot continue the quoted string to a second line.

```
"This is Unicode Data."
```

- DWORD: A 32-bit number

The data following the DWORD keyword is interpreted as a single, 32-bit value. It can be entered as a single value or as a list of values, separated by commas, with the least significant value first, provided the total is 32 or fewer bits.

For example, the following valid specifications are equivalent:

```
dword:44332211   DWORD:11,22,33,44   Dword:2211,4433
```

The following specifications are invalid because the total always exceeds 32 bits:

```
dword:5544332211
dword:11,22,33,44,55
dword:2211,554433
```

- EXPAND_SZ: A string of Unicode characters. The data following the EXPAND_SZ keyword is interpreted as a list of 4-byte Unicode values. For example:

  ```
  expand_sz:43,44,45
  ```

  This example stores the Unicode string "CDE".

- MULTI_SZ: A concatenated array of SZ strings. The data following the MULTI_SZ keyword is interpreted as a list of 4-byte Unicode values, specifying two or more terminated SZ strings. For example:

  ```
  MULTI_SZ:52,61,69,6e,00,53,6c,65,65,74,00,53,6e,6f,77,00
  ```

  This example stores the Unicode strings "Rain", "Sleet", and "Snow".

- HEX: Binary data. The data following the HEX keyword is interpreted as a list of hex values. For example:

  ```
  HEX:0F,C0,F0,FF
  ```

- DEC: Binary data. The data following the DEC keyword is interpreted as a list of decimal values. For example:

  ```
  DEC:15,192,240,255
  ```

- OCT: Binary data. The data following the OCT keyword is interpreted as a list of octal values. For example:

  ```
  OCT:17,300,360,377
  ```

- BIN: Binary data. The data following the BIN keyword is interpreted as a list of binary values. For example:

  ```
  BIN:1111,11000000,11110000,11111111
  ```

NOTES

– The input format is similar to the IMPORT and EXPORT file format.

– The input data type is independent of the data storage type, which is specified by the /TYPE_CODE qualifier.

– When you specify a binary input type (HEX, DEC, OCT, or BIN) for each value in the record, the data is stored in the fewest bytes possible. No alignment is performed. To ensure proper alignment, always enter any list of values as byte values. This input should be a stream of byte values, with the least significant byte first. For example:

```
DEC:253,    254,    255,    256, 257   (bytes/words, unaligned)
DEC:253, 0, 254, 0, 255, 0, 256, 257   (bytes/words, word aligned)
DEC:253, 0, 254, 0, 255, 0, 0, 1, 0, 2 (byte stream, word aligned)
```

– The input and storage types must be compatible. For example, you cannot specify value type DWORD (which means that the stored data is 4 bytes) and then input the SZ string "A" because then the SZ string would consist of the character 00000041 and the terminator 00000000, which cannot fit in a DWORD. Attempting this will result in a REG-E-INVDATA error. However, specifying an empty string ("") does work and stores just the terminator. Conversely, the input type DWORD specifies an input data length of 4 bytes. You cannot specify value type SZ and input type DWORD:00000041 because SZ strings must be terminated. The only valid DWORD you can enter in this case is 00000000.

– When specifying /INPUT, you cannot specify the /DATA qualifier.

/FLAGS=*flag*
Specifies the data flags value. This is an application-dependent 64-bit flag specified as a decimal number or as a hexadecimal number preceded by 0x or %X.

/LINK=(TYPE=*value*, NAME=*key-name*)
Defines the key as a link to another key. The link value must be one of the following:

- SYMBOLICLINK
- NONE

To remove a link, enter the following:

```
/LINK=(TYPE=NONE,NAME="")
```

/NAME=*string*
Specifies the name of the new value.

/TYPE_CODE=*type*
Specifies the type of the new value. The type value must be one of the following:

- SZ: a null-terminated Unicode string
- EXPAND_SZ: a string of Unicode characters
- MULTI_SZ: a concatenated array of SZ strings
- DWORD: a 32-bit number
- BINARY: raw binary data

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> CREATE VALUE/DATA=COSMOS/TYPE=SZ/NAME=COMPUTERNAME
HKEY_LOCAL_MACHINE\NODE
```

Creates the COMPUTERNAME value for the key HKEY_LOCAL_MACHINE\NODE and sets its type to SZ and its data value to COSMOS.

```
REG> CREATE VALUE
HKEY_USERS\CVEX1/NAME=SZ-HEX/TYPE=SZ/INPUT=SYS$INPUT
HEX:41,00,00,00,42,00,00,00,43,00,00,00,44,00,00,00,45,00,00,00,\
46,00,00,00,00,00,00,00
```

This example creates the SZ-HEX value for the key HKEY_USERS\CVEX1 and sets its type to SZ. The data is entered as hex data. In this particular case, the data equates to the Unicode string "ABCDEF". It would be simpler to enter ABCDEF as a string. However, this format provides the capability of entering any Unicode value, including those you may not be able to input directly as a string.

```
REG> CREATE POWERS2.DAT
DEC:2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,\
131072,262144,524288,1048576,2097152,4194304,8388608,16777216,\
33554432,67108864,134217728,268435456,536870912,1073741824

REG> CREATE VALUE -
HKEY_USERS\CVEX2/NAME=BIN-FILE/TYPE=BINARY/INPUT=POWERS2.DAT
```

This example creates data file POWERS2.DAT containing a series of decimal values. Next the user creates the value BIN-FILE for the key HKEY_USERS\CVEX2 and sets its type to BINARY. The raw binary data is read from file POWERS2.DAT to BIN-FILE.

DELETE KEY

Removes a specified key from the OpenVMS Registry database. The system does not delete a key if the key has subkeys.

Caution

Deleting a key results in symbolic links not being followed. This is because the system deletes the key you specify, even if it has symbolic links.

Note

The OpenVMS Registry database predefined keys are reserved keys and cannot be deleted. These keys include HKEY_USER, HKEY_LOCAL_MACHINE, and HKEY_CLASSES_ROOT. For a complete list, see Section 12.3.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

DELETE KEY key-path key-name

Parameters

key-path
Specifies the key path.

key-name
Specifies the name of the key to delete.

Qualifiers

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> DELETE KEY HKEY_USERS\NODE GUEST
```

Deletes the GUEST key from the OpenVMS Registry database.

DELETE TREE

Removes the specified key and all of its subkeys from the OpenVMS Registry database.

Caution

Deleting a key results in symbolic links not being followed. This is because the system deletes the key you specify, even if it has symbolic links.

Note

The OpenVMS Registry database predefined keys are reserved keys and cannot be deleted. These keys include HKEY_USER, HKEY_LOCAL_MACHINE, and HKEY_CLASSES_ROOT. For a complete list, see Section 12.3.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

DELETE TREE key-path key-name

Parameters

key-path
Specifies the key path.

key-name
Specifies the name of the top level key of the tree to be deleted.

Qualifiers

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG>  CREATE KEY HKEY_USERS\NODE\GUEST
REG>  CREATE KEY HKEY_USERS\NODE\GUEST\SUBKEY1
REG>  CREATE KEY HKEY_USERS\NODE\GUEST\SUBKEY2
REG>  CREATE KEY HKEY_USERS\NODE\GUEST\SUBKEY1\SUBKEY1_2
REG>  DELETE TREE HKEY_USERS\NODE GUEST
```

Deletes the GUEST key and its subkeys SUBKEY1, SUBKEY2, and SUBKEY1\SUBKEY1_2 from the OpenVMS Registry database.

DELETE VALUE

Removes a value from a specified key.

Caution

Deleting a value results in symbolic links not being followed. This is because the system deletes the value you specify, even if it has symbolic links.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

DELETE VALUE key-name value-name

Parameters

key-name
Specifies the key name whose value should be removed.

value-name
Specifies the value to remove.

Qualifiers

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> DELETE VALUE HKEY_USERS\GUEST PASSWORD
```

Deletes the PASSWORD value from the GUEST key.

EXIT

Enables you to exit from REG$CP and return to the DCL command prompt. You can also return to the DCL command level by pressing Ctrl/Z.

Format

EXIT

EXPORT DATABASE

Exports the entire OpenVMS Registry database contents to a text format file.

The default file format is OpenVMS format. You can also specify that the file be exported as a Windows NT-compatible text file format, which you can use to import key names and values into a Windows Registry.

This command requires the SYSPRV privilege.

Format

```
EXPORT DATABASE
```

Qualifiers

/FORMAT=[NT | VMS]
Specifies the format in which the system writes the database. VMS is the default.

If you intend to import the exported text file to an OpenVMS system, do not specify /FORMAT=NT. NT Export Registry File format does not preserve OpenVMS specific data structures.

/LOG
Displays the export progress to the screen.

/OUTPUT=*file-name*
Specifies a name for the exported file. The default output file name is `REGISTRY.TXT`.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> EXPORT DATABASE/LOG/OUTPUT=TUES_VERSION.TXT/FORMAT=VMS
```

The `EXPORT` command in this example logs the progress of the export to the screen as the system exports the entire OpenVMS Registry database to the `TUES_VERSION.TXT` file in VMS format.

EXPORT KEY

Exports a specific OpenVMS Registry database key (and optionally its subkeys) to a text format file. NOSUBKEYS is the default.

The default file format is OpenVMS format. You can also specify that the file be exported as a Windows NT-compatible text file format, which you can use to import key names and values into a Windows Registry.

This command requires the SYSPRV privilege.

Format

EXPORT KEY *key-name*

Parameters

DATABASE
Exports the full OpenVMS Registry database.

KEY [*key-name* [/[NO]SUBKEYS]]
Exports a specific OpenVMS Registry key and, optionally, its subkeys. NOSUBKEYS is the default.

Qualifiers

/FORMAT=[NT | VMS]
Specifies the format in which the system writes the database. VMS is the default.

Note

If you intend to import the exported text file to an OpenVMS system, do not specify /FORMAT=NT. NT Export Registry File format does not preserve OpenVMS specific data structures.

/LOG
Displays the export progress to the screen.

/OUTPUT=*file-name*
Specifies a name for the exported file. The default output file name is `REGISTRY.TXT`.

/SUBKEY
/NOSUBKEYS)
Specifies whether or not the key's subkeys are also to be exported. The default is /NOSUBKEYS.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

HELP

Displays information about using the REG$CP utility and includes formats and explanations for commands, parameters, and qualifiers.

Format

HELP [*keyword* [...]]

Parameters

keyword [...]
Specifies one or more keywords for a command and its subtopics.

IMPORT

Allows a user to import key definitions from a text format file (created by the EXPORT command) into an OpenVMS Registry database.

You can also import a Windows NT Export Registry File into an OpenVMS Registry database.

Conversion of Windows Resource Descriptors

You can import Windows resource descriptors into the OpenVMS Registry database, even though OpenVMS does not support NT resource descriptors. Typically, you import keys from a file created using the EXPORT/FORMAT=VMS command. The system displays a message when importing NT resource descriptors and converts them to the binary data type.

This command requires the SYSPRV privilege.

Format

IMPORT

Parameters

None

Qualifiers

/INPUT=*file-name*
Specifies a name of the file to import. The default input file name is `REGISTRY.TXT`.

/LOG
Displays the import progress to the screen.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> IMPORT/LOG/INPUT=TUES_VERSION.TXT
```

The IMPORT command in this example logs the progress of the import to the screen as the system imports the TUES_VERSION.TXT file.

LIST KEY

Displays the attributes for the specified key.

Note

Symbolic links are not followed.

This command requires the SYSPRV privilege or the REG$LOOKUP rights identifier.

Format

LIST KEY key-name

Parameters

key-name
Specifies the name of the key to list.

Qualifiers

/CACHE_ACTION
Displays the cache attribute for the key.

/CLASS_NAME
Displays the class name of the subkey.

/FULL
Displays all available information—that is, information displayed by the /LAST_WRITE, /CACHE_ACTION, /INFORMATION, /LINK_PATH, and /CLASS_NAME qualifiers.

/INFORMATION
Displays the information (subkey number, value number, subkey name max, and so on) about the specified key.

/LAST_WRITE
Displays the time when the subkey was last updated.

/LINK_PATH
Displays the key path to which the subkey is linked.

/OUTPUT=*file-spec*
Controls where the output of the command is sent. If you do not specify a file name, the system uses the default name REGISTRY.LIS.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> LIST KEY/FULL HKEY_USERS\GUEST

Key name:            HKEY_USERS\GUEST
Security policy:     REG$K_POLICY_NT_40
Volatile:            REG$K_NONE
Cache:               REG$K_WRITEBEHIND
Class:               System Authorization
Link Type:           REG$K_NONE
Last written:         7-AUG-1998 12:42:08.55

Key information:
  Number of subkeys:             2        Number of values: 0
  Max size of subkey name:      40        Max size of class name: 40
  Max size of value name:        0        Max size of value data: 0

Subkey(s):

    Key name:            QUOTAS
    Security policy:     REG$K_POLICY_NT_40
    Volatile:            REG$K_NONE
    Cache:               REG$K_WRITEBEHIND
    Class:               Disk quota
    Link Type:           REG$K_NONE
    Last written:         7-AUG-1998 12:41:19.21

    Key information:
      Number of subkeys:             0        Number of values: 0
      Max size of subkey name:       0        Max size of class name: 0
      Max size of value name:        0        Max size of value data: 0

    Key name:            IDENTIFIER
    Security policy:     REG$K_POLICY_NT_40
    Volatile:            REG$K_NONE
    Cache:               REG$K_WRITETHRU
    Class:               Disk quota
    Link Type:           REG$K_SYMBOLICLINK
    Link Path:           HKEY_LOCAL_MACHINE\SOFTWARE\IDENTIFIER\GUEST
    Last written:         7-AUG-1998 12:42:08.55

    Key information:
      Number of subkeys:             0        Number of values: 0
      Max size of subkey name:       0        Max size of class name: 0
      Max size of value name:        0        Max size of value data: 0
```

Note

The `Max sizes` information shows the number of bytes, not characters. (Each character is 4 bytes long.)

LIST SECURITYDESCRIPTOR

Displays the security descriptor associated with the specified key.

A security descriptor consists of a SECURITY_DESCRIPTOR structure and its associated security information. Security information can include security identifiers (SIDs), a system access-control list (SACL), and a discretionary access-control list (DACL).

This command requires the SYSPRV privilege or the REG$LOOKUP rights identifier.

Format

LIST SECURITYDESCRIPTOR key-name

Parameters

key-name
Specifies the name of the key whose security descriptor will be displayed.

Qualifiers

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

The following LIST SECURITYDESCRIPTOR command displays the security descriptor for the root key, HKEY_USERS.

```
REG> LIST SECURITYDESCRIPTOR HKEY_USERS

   Security Descriptor:

      Revision:              0x01
      Control:               0x8004 (SE_DACL_PRESENT,
                                     SE_SELF_RELATIVE)
      Owner Sid:             S-1-5-20-220
      Group Sid:             S-1-5-20-220
      Dacl:
         Revision:           0x02
         Size:               0x0048
         Ace Count:          0x0003

         Ace #1:
            Type:            0x00 (ACCESS_ALLOWED_ACE_TYPE)
            Flags:           0x03 (OBJECT_INHERIT_ACE,
                                   CONTAINER_INHERIT_ACE)
            Size:            0x0018
            Access Mask:     0x000f003f (Full Control)
            Sid:             S-1-5-20-220
```

```
   Ace #2:
      Type:            0x00 (ACCESS_ALLOWED_ACE_TYPE)
      Flags:           0x03 (OBJECT_INHERIT_ACE,
                             CONTAINER_INHERIT_ACE)
      Size:            0x0014
      Access Mask:     0x00020019 (Query Value, Enumerate
                                   Subkeys, Notify, Read Control)
      Sid:             S-1-1-0 (World)

   Ace #3:
      Type:            0x00 (ACCESS_ALLOWED_ACE_TYPE)
      Flags:           0x03 (OBJECT_INHERIT_ACE,
                             CONTAINER_INHERIT_ACE)
      Size:            0x0014
      Access Mask:     0x000f003f (Full Control)
      Sid:             S-1-5-12 (System)
```

The command in the following example displays the security descriptor for the HKEY_LOCAL_MACHINE\SOFTWARE key.

```
REG> LIST SECURITYDESCRIPTOR HKEY_LOCAL_MACHINE\SOFTWARE

  Security Descriptor:

     Revision:           0x01
     Control:            0x8004 (SE_DACL_PRESENT,
                                 SE_SELF_RELATIVE)
     Owner Sid:          S-1-5-20-220
     Group Sid:          S-1-5-20-220
     Dacl:
        Revision:        0x02
        Size:            0x005c
        Ace Count:       0x0004

        Ace #1:
           Type:         0x00 (ACCESS_ALLOWED_ACE_TYPE)
           Flags:        0x03 (OBJECT_INHERIT_ACE,
                               CONTAINER_INHERIT_ACE)
           Size:         0x0018
           Access Mask:  0x000f003f (Full Control)
           Sid:          S-1-5-20-220

        Ace #2:
           Type:         0x00 (ACCESS_ALLOWED_ACE_TYPE)
           Flags:        0x03 (OBJECT_INHERIT_ACE,
                               CONTAINER_INHERIT_ACE)
           Size:         0x0014
           Access Mask:  0x000f003f (Full Control)
           Sid:          S-1-3-0

        Ace #3:
           Type:         0x00 (ACCESS_ALLOWED_ACE_TYPE)
           Flags:        0x03 (OBJECT_INHERIT_ACE,
                               CONTAINER_INHERIT_ACE)
           Size:         0x0014
           Access Mask:  0x0003001f (Query Value, Set Value,
                                     Create Subkey, Enumerate
                                     Subkeys, Notify, Delete,
                                     Read Control)
           Sid:          S-1-1-0 (World)

        Ace #4:
           Type:         0x00 (ACCESS_ALLOWED_ACE_TYPE)
           Flags:        0x03 (OBJECT_INHERIT_ACE,
                               CONTAINER_INHERIT_ACE)
           Size:         0x0014
           Access Mask:  0x000f003f (Full Control)
           Sid:          S-1-5-12 (System)
```

LIST VALUE

Displays all values and value attributes of the specified key.

Note

Symbolic links are not followed.

This command requires the SYSPRV privilege or the REG$LOOKUP rights identifier.

Format

LIST VALUE key-name

Parameters

key-name
Specifies the name of the key to enumerate.

Qualifiers

/DATA
Displays an ASCII representation of the value in hexadecimal format.

/FLAGS
Displays an ASCII representation of the data flag of the value in hexadecimal format.

/FULL
Displays all available information—that is, information displayed by the /TYPE_CODE, /LINK_PATH, /DATA_FLAGS, and /VALUE_DATA qualifiers.

/LINK_PATH
Displays the key path to which the subkey is linked.

/OUTPUT=*file-spec*
Controls where the output of the command is sent. If you do not specify a file name, the system uses the default name REGISTRY.LIS.

/TYPE_CODE
Display the type code of the value.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> LIST VALUE/TYPE_CODE/DATA HKEY_LOCAL_MACHINE\SOFTWARE\FORTRAN

   Key name:            HKEY_LOCAL_MACHINE\SOFTWARE\FORTRAN
   Security policy:     REG$K_POLICY_NT_40
   Volatile:            REG$K_NONE
   Last written:        11-AUG-1998 16:27:55.81

   Value(s):

     Value name:   Version
     Volatile:     REG$K_NONE
     Type:         REG$K_SZ
     Data:         5.3-50

     Value name:   Date Installed
     Volatile:     REG$K_NONE
     Type:         REG$K_SZ
     Data:         04-Jan-1998
```

The `LIST VALUE/TYPE_CODE/DATA` command in this example displays the `FORTRAN` key and its value names, types, and data.

MODIFY KEY

Modifies the attributes of the specified key.

Caution

Modifying a key results in symbolic links not being followed. This is because the system modifies the key you specify, not the key pointed to by the symbolic link.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

MODIFY KEY key-name

Parameters

key-name
Specifies the name of the key to modify.

Qualifiers

/CACHE_ACTION=*value*
Specifies the cache attribute for the new key. The *value* can be WRITEBEHIND (write to disk later) or WRITETHRU (write to disk immediately).

If you omit /CACHE_ACTION, the system creates the key with the cache attribute set to REG$K_WRITEBEHIND.

/CLASS_NAME=*string*
Specifies the new class name of the key.

/LINK=(TYPE=3D*value*, NAME=*key-name*)
Defines the key as a link to another key. The link value must be one of the following:

- SYMBOLICLINK
- NONE

To remove a link, enter the following:

```
/LINK=(TYPE=NONE,NAME="")
```

/NEW_NAME=*new-key-name*
Specifies the new name of the key.

/SECPOLICY=*policy*
Defines the security policy for the key. Currently the only valid policy is NT_40.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default

interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> MODIFY KEY/CACHE_ACTION=WRITEBEHIND
HKEY_USERS\GUEST
```

Modifies the cache attribute of the GUEST key.

MODIFY TREE

Modifies the information for the specified key and its subkeys.

Caution

Modifying a tree results in symbolic links not being followed. This is because the key and subkeys you specify are modified, not the key pointed to by the symbolic link.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

MODIFY TREE key-name

Parameters

key-name
Specifies the name of the key to modify.

Qualifiers

/CACHE_ACTION=*value*
Specifies the cache attribute for the key and its subkeys. The value can be WRITEBEHIND (write to disk later) or WRITETHRU (write to disk immediately).

/CLASS_NAME=*string*
Specifies the new class name for the given key and all its subkeys.

/SECPOLICY=*policy*
Defines the security policy for the key. Currently the only valid policy is NT_40.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> MODIFY  TREE/CACHE_ACTION=WRITEBEHIND
HKEY_USERS\GUEST
```

Modifies the cache attribute of the GUEST key and all its subkeys.

MODIFY VALUE

Specifies the data component for the specified value. This command modifies an existing value.

Caution

Modifying a value results in symbolic links not being followed. This is because the system modifies the value you specified, not the value pointed to by the symbolic link.

This command requires the SYSPRV privilege or the REG$UPDATE rights identifier.

Format

MODIFY VALUE /NAME=*string* key-name

Parameters

key-name
Specifies the name of the key for which to set the value.

Qualifiers

/DATA=*value*
Specifies the data for the value. The value can be:

- A string (for example, /DATA=COSMOS)
- An array of strings separated by a comma and enclosed in parentheses (for example, /DATA=(COSMOS,Noidea)
- A longword in binary, octal, decimal, or hexadecimal format. %B, %O, %D, and %X, or 0B, 0O, 0D, and 0X prefixes specify the format. The default is decimal.

  Examples:

```
/DATA=%X1A0FCB (hex)
/DATA=0X1A0FCB (hex)
/DATA=%D1234 (decimal)
/DATA=1234 (decimal, by default)
```

Note

You cannot specify the /INPUT qualifier with /DATA.

/FLAGS=*flag*
Specifies the data flags value. This is an application-dependent 64-bit flag specified as a decimal number or as a hexadecimal number preceded by 0x or %X.

/INPUT=*filename*
Specifies that the value data is to be read from a file.

The input value data can be specified in one of the following formats. With the exception of SZ, the format is specified by a keyword at the start of the file. The keyword can be entered in uppercase or lowercase. Input records following a keyword can span multiple lines; use "\" at the end of any continuation lines.

- SZ: a null-terminated Unicode string

  Enter SZ data by enclosing the record in quotes. You do not specify an SZ keyword for this input value type. You cannot continue the quoted string to a second line.

  ```
  "This is Unicode Data."
  ```

- DWORD: A 32-bit number.

  The data following the DWORD keyword is interpreted as a single, 32-bit value. It can be entered as a single value or as a list of values, separated by commas, with the least significant value first, provided the total is 32 or fewer bits.

  For example, the following valid specifications are equivalent:

  ```
  dword:44332211   DWORD:11,22,33,44   Dword:2211,4433
  ```

  The following specifications are invalid because the total always exceeds 32 bits:

  ```
  dword:5544332211
  dword:11,22,33,44,55
  dword:2211,554433
  ```

- EXPAND_SZ: A string of Unicode characters. The data following the EXPAND_SZ keyword is interpreted as a list of 4-byte Unicode values. For example:

  ```
  expand_sz:43,44,45
  ```

  This example stores the Unicode string "CDE".

- MULTI_SZ: A concatenated array of SZ strings. The data following the MULTI_SZ keyword is interpreted as a list of 4-byte Unicode values, specifying two or more terminated SZ strings. For example:

  ```
  MULTI_SZ:52,61,69,6e,00,53,6c,65,65,74,00,53,6e,6f,77,00
  ```

  This example stores the Unicode strings "Rain", "Sleet", and "Snow".

- HEX: Binary data. The data following the HEX keyword is interpreted as a list of hex values. For example:

  ```
  HEX:0F,C0,F0,FF
  ```

- DEC: Binary data. The data following the DEC keyword is interpreted as a list of decimal values. For example:

  ```
  DEC:15,192,240,255
  ```

- OCT: Binary data. The data following the OCT keyword is interpreted as a list of octal values. For example:

  ```
  OCT:17,300,360,377
  ```

- BIN: Binary data. The data following the BIN keyword is interpreted as a list of binary values. For example:

  ```
  BIN:1111,11000000,11110000,11111111
  ```

NOTES

- The input format is similar to the IMPORT and EXPORT file format.
- The input data type is independent of the data storage type, which is specified by the /TYPE_CODE qualifier.
- When you specify a binary input type (HEX, DEC, OCT or BIN) for each value in the record, the data is stored in the fewest bytes possible. No alignment is performed. To insure proper alignment, always enter any list of values as byte values. This input should be a stream of byte values, with the least significant byte first. For example:

```
DEC:253,    254,    255,    256, 257   (bytes/words, unaligned)
DEC:253, 0, 254, 0, 255, 0, 256, 257   (bytes/words, word aligned)
DEC:253, 0, 254, 0, 255, 0, 0, 1, 0, 2 (byte stream, word aligned)
```

- The input and storage types must be compatible. For example, you cannot specify value type DWORD (which means that the stored data is 4 bytes) and then input the SZ string "A" because then the SZ string would consist of the character 00000041 and the terminator 00000000, which cannot fit in a DWORD. Attempting this will result in a REG-E-INVDATA error. However, specifying an empty string ("") does work and stores just the terminator. Conversely, the input type DWORD specifies an input data length of 4 bytes. You cannot specify value type SZ and input type DWORD:00000041 because SZ strings must be terminated. The only valid DWORD you can enter in this case is 00000000.
- When specifying /INPUT, you cannot specify the /DATA qualifier.

/LINK=(TYPE=*value*, NAME=*key-name*)
Defines the key as a link to another key. The link value must be one of the following:

- SYMBOLICLINK
- NONE

To remove a link, enter the following:

```
/LINK=(TYPE=NONE,NAME="")
```

/NAME=*string*
Specifies the name of the value.

/TYPE_CODE=*type*
Specifies the type of the new value. The type value must be one of the following:

- SZ: a null-terminated Unicode string
- EXPAND_SZ: a string of Unicode characters
- MULTI_SZ: a concatenated array of SZ strings
- DWORD: a 32-bit number
- BINARY: raw binary data

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default

interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

/WRITEBEHIND
/NOWRITEBEHIND (default)
Specifies when the information can be written to disk. /WRITEBEHIND specifies that the information can be written to disk later. /NOWRITEBEHIND specifies write-through operation (that is, the information must be written to disk immediately).

Examples

```
REG> MODIFY VALUE/DATA=COSMOS/TYPE=SZ/NAME=COMPUTERNAME
HKEY_LOCAL_MACHINE\NODE
```

Creates `COMPUTERNAME` value for the key `HKEY_LOCAL_MACHINE\NODE`, and sets its type code to `SZ` and its data value to `COSMOS`.

SEARCH KEY

Displays the path name of all the keys that match the specified key.

This command requires the SYSPRV privilege or the REG$LOOKUP rights identifier.

Format

SEARCH KEY key-search

Parameters

key-search
Specifies the key name for which to search.

Qualifiers

/OUTPUT=*file-spec*
Controls where the output of the command is sent. If you do not specify a file name, the system uses the default name REGISTRY.LIS.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> SEARCH KEY
HKEY_LOCAL_MACHINE\...\NODE
HARDWARE\CLUSTER\NODE
HARDWARE\LOCAL\NODE
NODE
```

Displays all the key paths that match the HKEY_LOCAL_MACHINE\...\NODE selection. The ellipsis (...) wildcard specifies that there can be any number of subkeys between the HKEY_LOCAL_MACHINE entry point and the NODE subkey. Note that the search is not case sensitive.

SEARCH VALUE

Displays the path name of all the values that match the specified value name.

This command requires the SYSPRV privilege or the REG$LOOKUP rights identifier.

Format

SEARCH VALUE key-name value-name

Parameters

key-name
Specifies the name of the key path to search.

value-name
Specifies the value name for which to search.

Qualifiers

/OUTPUT=*file-spec*
Controls where the output of the command is sent. If you do not specify a file name, the system uses the default name REGISTRY.LIS.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> SEARCH VALUE HKEY_LOCAL_MACHINE\... *AM%
HARDWARE\CLUSTER\Name
HARDWARE\CLUSTER\NODE\Name
HARDWARE\LOCAL\NODE\Name
NODE\COMPUTERNAME
```

Displays all the value names that match the HKEY_LOCAL_MACHINE\...*am% selection. The ellipsis (...) wildcard specifies that there can be any number of subkeys between the HKEY_LOCAL_MACHINE entry point and the *am% value name. Note that the search is not case sensitive.

SHOW

Displays OpenVMS Registry server internal statistics and information.

- **SHOW COUNTERS**

  Displays monitoring information from the OpenVMS Registry server.

- **SHOW FILE**

  Displays status information on files loaded into the OpenVMS Registry server.

This command requires the SYSPRV privilege or the REG$PERFORMANCE rights identifier.

Format

SHOW COUNTERS/FILE [name]

SHOW FILE [name]

Parameters

name
Identifies the file (used with the /FILE qualifier only).

Qualifiers

/FILE
Displays counters for the specified file or for all files.

/PERFORMANCE
Displays performance counters.

/OUTPUT=*file-spec*
Controls where the output of the command is sent. If you do not specify a file name, the system uses the default name REGISTRY.LIS.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> SHOW COUNTERS/FILE
```

Displays monitoring information from the OpenVMS Registry server.

```
REG> SHOW COUNTERS/FILE REGISTRY$USERS
```

Displays monitoring information for file REGISTRY$USERS from the OpenVMS Registry server.

SPAWN

Creates a subprocess of the current process. Portions of the current process context are copied to the subprocess. You can use the SPAWN command to temporarily leave REG$CP, perform other functions, then return to REG$CP.

Format

SPAWN [command-string]

Parameters

command-string
Command to be executed.

START MONITORING

Starts a monitoring component within the OpenVMS Registry server.

This command requires the SYSPRV privilege or the REG$PERFORMANCE rights identifier.

Format

START MONITORING/FILE [name]

START MONITORING/PERFORMANCE

Parameters

name
Identifies the file (used with the /FILE qualifier only).

Qualifiers

/FILE
Start gathering counters for the specified file or for all files.

/PERFORMANCE
Start gathering performance counters.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> START MONITORING/PERFORMANCE
```

Enables a monitoring component of the OpenVMS Registry.

STOP MONITORING

Stops a monitoring component within the OpenVMS Registry server.

This command is used to stop a monitoring component within the OpenVMS Registry server.

This command requires the SYSPRV privilege or the REG$PERFORMANCE rights identifier.

Format

STOP MONITORING/FILE [name]

STOP MONITORING/PERFORMANCE

Parameters

name
Identifies the file (used with the /FILE qualifier only).

Qualifiers

/FILE
Stop gathering counters for the specified file or for all files.

/PERFORMANCE
Stop gathering performance counters.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> STOP MONITORING/PERFORMANCE
```

Disables a monitoring component of the OpenVMS Registry.

WAIT

Waits for the specified number of seconds or waits for a random number of seconds within a range of seconds for command completion.

Format

WAIT [seconds]

WAIT [minimum-seconds maximum-seconds]

Parameters

seconds
Specifies the number of seconds that you are willing to wait.

minimum-seconds
Specifies the minimum amount of time, in seconds, that you are willing to wait.

maximum-seconds
Specifies the maximum amount of time, in seconds, that you are willing to wait.

ZERO COUNTERS

Initializes counters within the OpenVMS Registry server.

This command requires the SYSPRV privilege or the REG$PERFORMANCE rights identifier.

Format

ZERO COUNTERS/FILE [name]

ZERO COUNTERS/PERFORMANCE

Parameters

name
Identifies the file (used with the /FILE qualifier only).

Qualifiers

/FILE
Initializes the file counters for the specified file or for all files.

/PERFORMANCE
Initializes all performance counters.

/WAIT=*seconds*
/NOWAIT
Specifies the maximum amount of time, in seconds, that you are willing to wait for command completion. If the Registry server does not complete the request in the specified interval, REG$CP returns REG-F-NORESPONSE. The default interval is 90 seconds. /NOWAIT is equivalent to specifying /WAIT=0, but there may still be a short wait period.

Examples

```
REG> ZERO COUNTERS/PERFORMANCE
```

This example resets the performance counters.

Part III

OpenVMS Events

This part contains reference information about OpenVMS Events.

15
OpenVMS Events

15.1 What are Events?

On a Windows system, an **event** is any significant occurrence in the system or an application—for example, a service starting or stopping, a user logging on or off, or accessing resources. When the system encounters an event, the **Event Log service** writes the event (or audit entry) in the form of a record that contains date and time, source, category, event number, user, and computer information to a system, security, or application log, creating an audit trail. On Windows systems, you display these logs and their recorded events using the **Event Viewer**.

With COM Version 1.1-B and higher for OpenVMS, OpenVMS supports both Windows logging and HP Advanced Server for OpenVMS logging of COM for OpenVMS events, and writes all COM for OpenVMS events to the DCOM$EVENTLOG.RPT text file. You can log a COM for OpenVMS event (such as the starting of a COM server on OpenVMS), and review these OpenVMS events from a Windows system or an OpenVMS system.

For a detailed review of OpenVMS Events dependencies and a description of how OpenVMS Events interacts with other parts of the OpenVMS infrastructure, see Section 4.10.

15.1.1 Suggested Reading

The following sources can provide you with more information on Events and related topics:

- Third-party books on event logging:
  - — *Windows Server NT 4.0 Unleashed*, Jason Garms, SAMS Publishing, Indianapolis, IN, 1998. ISBN: 0-672-30933-5.
  - — *Win32 System Services: The Heart of Windows 95 and Windows NT*, Marshall Brain, Prentice Hall, Upper Saddle River, NJ, 1996. ISBN: 0-13-324732-5.
- Other sources:
  - — Microsoft *Win32 Software Development Kit*

    In particular, see the sections about the `RegisterEventSource`, `ReportEvent`, and `DeregisterEventSource` functions, and System Services: *Event Logging* section.

15.2 Overview of OpenVMS Events

The system logs OpenVMS Events to a Windows event log, to the HP Advanced Server for OpenVMS event log, and to a log file on the OpenVMS system.

You can use the following techniques to view OpenVMS Events:

- Windows event viewer (see Section 15.2.1)
- HP Advanced Server for OpenVMS event viewer (see Section 15.2.2)
- OpenVMS event log file (see Section 15.2.3)

15.2.1 Viewing OpenVMS Events Using Windows Event Viewer

Use the following procedure to view OpenVMS Events through the Windows event viewer:

1. Start the Windows event viewer.

   From the **Start** menu, select **Programs**, **Administrative Tools**, **Event Viewer**.

2. From the Event Viewer window, click the menu bar **Log** option. Click **Select Computer....**, and select the OpenVMS system from the list box.

3. From the Event Viewer window, click the menu bar **Log** option. Click **System** to display the System event log. The System event log contains the COM for OpenVMS events.

   To display COM for OpenVMS events only, use the following procedure:

   - From the Event Viewer window, click the menu bar **View** option. Click **Filter Events...**.

     The system displays the Filter window.

   - On the Filter window, click the Source: list box. From the list, choose DCOM.

15.2.2 Viewing OpenVMS Events Using HP Advanced Server for OpenVMS Event Viewer

Use the following procedure to view the COM for OpenVMS events:

1. Ensure that the HP Advanced Server for OpenVMS is running.

2. Enter the following HP Advanced Server for OpenVMS ADMINISTRATOR command:

   ```
   $ ADMIN SHOW EVENTS/TYPE=SYSTEM/SOURCE=DCOM/FULL
   ```

   The viewer displays COM for OpenVMS events only, along with any additional information associated with the COM for OpenVMS event.

15.2.3 Event Logging on OpenVMS Only

In some cases, you might want to write and view COM for OpenVMS events only on an OpenVMS system. In place of the Windows log, HP has included an alternate event logger that writes COM event information to an OpenVMS file. You can find this file in the following location:

```
SYS$MANAGER:DCOM$EVENTLOG.RPT
```

COM for OpenVMS creates this event logging report automatically when the COM server (DCOM$RPCSS) encounters an error. The event logger appends new events at the bottom (end) of the file. A logged event has the following format:

```
event type : ddd mmm dd hh:mm:ss yyyy
First event message

event type : ddd mmm dd hh:mm:ss yyyy
Second event message
.
.
.
```

Example 15–1 shows the contents of an event log.

Example 15–1 Sample OpenVMS Event Log

```
$ Type SYS$MANAGER:DCOM$EVENTLOG.RPT

❶
ERROR : Tue Sep 15 11:18:54 1998
Unable to start a DCOM Server: {5E9DDEC7-5767-11CF-BEAB-00AA006C3606}
Runas (null)/SMITH
The Windows error: 1326
Happened while starting: device:[account]SSERVER.EXE

❷
ERROR : Tue Sep 15 19:14:45 1998
The server {0C092C21-882C-11CF-A6BB-0080C7B2D682} did not register
with DCOM within the required timeout.
```

❶ The system logged the first error event on Tue Sep 15 11:18:54 1998. The COM server (DCOM$RPCSS) was unable to start the COM application *device*:[*account*]SSERVER.EXE on behalf of the client running under the SMITH account. (The client may have received an error such as "access denied.") The resulting Windows error was 1326, which translates as "Logon failure: unknown user name or bad password."

If you see this error, check the validity of the user account using the OpenVMS Authorize utility (AUTHORIZE).

❷ The system logged the second error event on Tue Sep 15 19:14:45 1998. The COM server (DCOM$RPCSS) was able to start the COM application {0C092C21-882C-11CF-A6BB-0080C7B2D682}, but the application did not run successfully. The application failed to register with DCOM$RPCSS within the specified time limit. (The client may have received an error such as "Server execution failed" CO_E_SERVER_EXEC_FAILURE.)

If you see this error, run the server application interactively to determine its integrity.

NTA$EVENTW
Interface to OpenVMS Events

Allows an application to record information in the event log files.

The NTA$EVENTW routine completes all operations synchronously.

Format

NTA$EVENTW [nullarg], func, itmlst, evsb

Arguments

nullarg

OpenVMS usage:	reserved
type:	longword (unsigned)
access:	read only
mechanism:	by value

Reserved for HP use.

func

OpenVMS usage:	function_code
type:	longword (unsigned)
access:	read only
mechanism:	by value

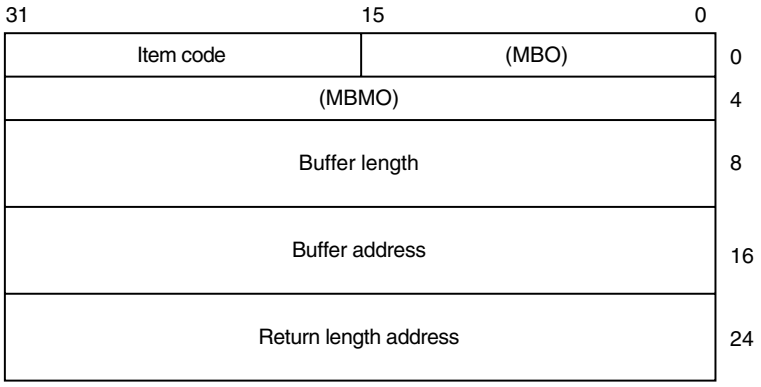
Function code specifying the function NTA$EVENTW is to perform. The func argument is a longword containing this function code. The $EVENTDEF macro defines the names of each function code.

itmlst

OpenVMS usage:	address of item list
type:	64-bit address
access:	read only
mechanism:	by value

Item list specifying information about the event source or the event. The itmlst argument is the 64-bit address of a list of item descriptors, each of which describes an item of information. An item list in 64-bit format is terminated by a quadword of 0.

The following diagram shows the 64-bit format of a single item descriptor.



evsb

OpenVMS usage:	address of status block
type:	64-bit address
access:	write only
mechanism:	by reference

Event status block to contain the completion status for the requested operation.

NTA$EVENTW sets the status block to 0 upon request initiation. Upon request completion, the EVT$L_VMS_STATUS field contains the primary (OpenVMS) completion status for the operation.

If an error occurs, EVT$L_NT_STATUS (if non-zero) is the secondary error status to further define the error condition.

Function Codes

EVT$_FC_REGISTER_EVENT_SOURCE
Open an association with an event log.

Item code	Required	Parameter	Data type
EVT$_SERVER_NAME	No	Input	String (4-byte Unicode)
EVT$_SOURCE	No	Input	String (4-byte Unicode)
EVT$_HANDLE	Yes	Output	Unsigned longword

- EVT$_SERVER_NAME

  The universal naming convention (UNC) name of the server on which this operation is to be performed.

  UNC names have the form *server**share**path**file*. This item must be zero or unspecified. This performs the operation on an available HP Advanced Server for OpenVMS server in the cluster.

- EVT$_SOURCE

  The name of the application that logs the event. This field associates an application message file that contains descriptive text with the application's event log entries.

If specified, the source must be a subkey of the Eventlog\System key, the Eventlog\Security key, or the Eventlog\Application registry key. For example, a source name of Myapp indicates a registry entry in the following:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\
   Services\Eventlog\Application\Myapp)
```

The Myapp registry value EventMessageFile names the path and message file to be used to translate this application's events.

The source can be unspecified or specified as NULL. In this case, the system logs events to the Application log file but the application logs no message file (and, as a result, no replacement text) for the associated events.

- EVT$_HANDLE

  Returns a handle to the Application event log. This handle is required input for other $EVENT functions.

  On failure, a handle of 0 is returned. This handle is outside the responsibility of the CloseHandle API.

EVT$_FC_REPORT_EVENT
Generate an event log entry.

Item code	Required	Parameter	Data type
EVT$_HANDLE	Yes	Input	Unsigned longword
EVT$_EVENT_TYPE	Yes	Input	Word mask
EVT$_EVENT_CATEGORY	No	Input	Word
EVT$_EVENT_ID	Yes	Input	Longword
EVT$_USER_SID	No	Input	NT Security ID
EVT$_NUMSTRINGS	No	Input	Word
EVT$_DATASIZE	No	Input	Longword
EVT$_STRING_ARRAY	No	Input	Array of varying-length descriptors. (4-byte Unicode)
EVT$_RAW_DATA	No	Input	Binary data

- EVT$_HANDLE

  Value returned by a previous EVT$_FC_REGISTER_EVENT_SOURCE call.

- EVT$_EVENT_TYPE

  Indicates the severity of the event. The type is one of the following:

  EVT$_SUCCESS
  EVT$_ERROR
  EVT$_WARNING
  EVT$_INFO
  EVT$_AUDIT_SUCCESS
  EVT$_AUDIT_FAILURE

  The severity type maps to its Windows equivalent, defined in WINNT.H.

- EVT$_EVENT_CATEGORY

  An integer value from 1 to 65535. EVT$_EVENT_CATEGORY is unique to a particular source.

EVT$_EVENT_CATEGORY allows an application to divide its message file into sections, each indexed by event ID. If you do not specify a category, the system defaults to a category of zero.

- EVT$_EVENT_ID

  An unlimited integer value. This value indexes the category in an application message file that locates the text string displayed for this event message. The event ID is unique to a particular source.

- EVT$_USER_SID

  The optional Windows Security ID of the thread logging the event. An application that has acquired Windows credentials through the $PERSONA system service can obtain its SID through calls to the `OpenProcessToken` and `GetTokenInformation` Win32 APIs. The format is opaque to this service.

- EVT$_NUMSTRINGS

  A count of the strings specified in the EVT$_STRING_ARRAY item code.

- EVT$_DATASIZE

  Length in bytes of the buffer indicated by the EVT$_RAW_DATA item code.

- EVT$_STRING_ARRAY

  An array of string pointers. Each entry points to a null terminated string. A description string in a message file can contain string placeholders in the form *%n*, where %1 indicates the first placeholder. Strings specified in this array replace these placeholders when the system displays the event message.

- EVT$_RAW_DATA

  Allows you to include binary data in an event message.

  For example, you might use this to dump a data structure from a failing component.

EVT$_DEREGISTER_EVENT_SOURCE
Close an association with an event log.

Item code	Required	Parameter	Data type
EVT$_HANDLE	Yes	Input	Unsigned longword

- EVT$_HANDLE

  Value returned by a previous EVT$_FC_REGISTER_EVENT_SOURCE call.

Item Codes

Item Code	Parameter Type	Data Type
EVT$_SERVER_NAME	Input	String
EVT$_SOURCE	Input	String
EVT$_HANDLE	Input/Output	Unsigned longword
EVT$_EVENT_TYPE	Input	Word mask

Item Code	Parameter Type	Data Type
EVT$_EVENT_CATEGORY	Input	Word
EVT$_EVENT_ID	Input	Longword
EVT$_USER_SID	Input	NT security ID
EVT$_NUMSTRINGS	Input	Word
EVT$_DATASIZE	Input	Longword
EVT$_STRING_ARRAY	Input	Array of string pointers
EVT$_RAW_DATA	Input	Binary data

Description

The NTA$EVENTW routine allows you to register and deregister an event source and report event data. This event logging allows you to record information from within an application. You can use the events routines to track progress within an application or identify problems encountered by an application.

The NTA$EVENTW routine completes synchronously; that is, control is returned to the caller only after the request completes.

Use the following process to write event data:

1. Register the event source.

   This operation defines the event log to which the system writes event data.

2. Report the event.

   This operation causes the system to write the information to the appropriate event log.

3. Deregister the event source.

   This operation frees resources acquired as part of the event source registration operation.

Condition Values Returned

SS$_NORMAL	Service completed successfully.
SS$_ACCVIO	One of the arguments cannot be read/written.
SS$_BADPARAM	Bad parameter.
SS$_NOPRIV	Insufficient privilege to access the specified event log.
SS$_TIMEOUT	Request timed out.
SS$_UNREACHABLE	Events service unavailable.
SS$_REJECT	The Windows LAN Manager server encountered an error. See the Win32 status for more information.

15.3 Writing Your Own Events

By default, the system logs DCOM events generated by COM for OpenVMS. In addition to recording COM for OpenVMS events, the system can also log COM application events for COM applications that you create.

The COM for OpenVMS kit includes sample code that shows how to generate an application event using Win32 APIs. You can use this example as is on a Windows system. The example also builds correctly using the instructions for building COM for OpenVMS applications on OpenVMS (to get the required header files from DCOM$LIBRARY). See Chapter 7 for these instructions. The example also includes the linking instructions to build the example using Wind/U.

15.4 Troubleshooting OpenVMS Events

Errors that occur during event reporting can be difficult to trace because of the number of intervening software layers through which the event passes. The following list describes how OpenVMS Events pass through other software layers until they are recorded in the Windows log.

1. An application calls one of the Win32 event functions (`RegisterEventSource`, `ReportEvent`, or `DeregisterEventSource`).
2. Using the supplied arguments, the Win32 API builds an appropriate item list and calls the NTA$EVENTW routine.
3. The NTA$EVENTW routine validates the information supplied (function code, item list, and so on) and builds an appropriate item list for the SYS$ACM system service.

   If NTA$EVENT detects any errors NTA$EVENT returns the errors to the Win32 API using R0 and the event status block.
4. The SYS$ACM system service validates the information and passes it to the NT ACME.

   If SYS$ACM detects any errors, SYS$ACM returns the errors to NTA$EVENTW using R0 and the ACM status block..
5. The NT ACME passes the supplied information (using an IPC pipe) to a dispatcher in the HP Advanced Server for OpenVMS.

   If the NT ACME detects any errors, the NT ACME returns the errors to the caller using the ACM status block.
6. The HP Advanced Server for OpenVMS dispatcher validates the information and calls the appropriate routines to perform the requested operation (register, report, or deregister).

   If the HP Advanced Server for OpenVMS detects any errors, it reports the errors to the NT ACME. The NT ACME passes the errors back to the other callers.

Checking the contents of the event status block help you determine where the failure might have happened. Table 15–1 lists (in order of importance) the checks you should perform.

Table 15–1 Troubleshooting OpenVMS Events Failures

R0 Status	Status Field Value	Component to Check
Failure (bit 0 clear)	EVT$L_NT_STATUS field is nonzero.	Error most likely occurred within HP Advanced Server for OpenVMS.
Failure	EVT$L_VMS_STATUS field is nonzero and the EVT$L_NT_STATUS is zero.	Error most likely occurred within the SYS$ACM system service or the NT ACME.
Failure	EVT$L_VMS_STATUS is zero and EVT$L_NT_STATUS is zero.	Error most likely occurred within the SYS$ACM system service.

Note

The Win32 API usually converts the error status to an appropriate NT error status code and makes it available through the GetLastError Win32 API. (The status returned by the event API simply indicates a generic failure.)

Part IV
Appendixes

This part contains reference information about COM for OpenVMS and the OpenVMS Registry.

The appendixes provide information about the MIDL compiler, troubleshooting tips, COM sample code, running COM for OpenVMS in an unauthenticated environment, and APIs and interfaces.

This part also includes a glossary and a list of acronyms.

A

MIDL Compiler Options

A.1 Mode

Switch	Use
/ms_ext	Microsoft extensions to the IDL language (default)
/c_ext	Allow Microsoft C extensions in the IDL file (default)
/osf	OSF mode - disables /ms_ext and /c_ext options
/app_config	Allow selected ACF attributes in the IDL file
/mktyplib203	MKTYPLIB Version 2.03 compatibility mode

A.2 Input

Switch	Use
/acf filename	Specify the attribute configuration file
/I directory-list	Specify one or more directories for include path
/no_def_idir	Ignore the current and the INCLUDE directories

A.3 Output File Generation

Switch	Use
/client none	Do not generate client files
/client stub	Generate client stub file only
/out directory	Specify destination directory for output files
/server none	Generate no server files
/server stub	Generate server stub file only
/syntax_check	Check syntax only; do not generate output files
/Zs	Check syntax only; do not generate output files
/old	Generate old format type libraries
/new	Generate new format type libraries

A.4 Output File Names

Switch	Use
/cstub filename	Specify client stub file name
/dlldata filename	Specify dlldata file name
/h filename	Specify header file name

Switch	Use
/header filename	Specify header file name
/iid filename	Specify interface UUID file name
/proxy filename	Specify proxy file name
/sstub filename	Specify server stub file name
/tlb filename	Specify type library file name

A.5 C Compiler and Preprocessor Options

Switch	Use
/cpp_cmd cmd_line	Specify name of C++ preprocessor
/cpp_opt options	Specify additional C++ preprocessor options
/D name[=def]	Pass #define name, optional value to C++ preprocessor
/no_cpp	Turn off the C++ preprocessing option
/nocpp	Turn off the C++ preprocessing option
/U name	Remove any previous definition (undefine)

A.6 Environment

Switch	Use
/char signed	C++ compiler default char type is signed
/char unsigned	C++ compiler default char type is unsigned
/char ascii7	Char values limited to 0-127
/dos	Target environment is MS-DOS client
/env dos	Target environment is MS-DOS client
/env mac	Target environment is Apple Macintosh
/env powermac	Target environment is Apple PowerMac
/env win16	Target environment is Microsoft Windows 16-bit (Win 3.x)
/env win32	Target environment is Microsoft Windows 32-bit (NT)
/mac	Target environment is Apple Macintosh
/ms_union	Use Midl 1.0 non-DCE wire layout for non-encapsulated unions
/oldnames	Do not mangle version number into names
/powermac	Target environment is Apple PowerMac
/rpcss	Automatically activate rpc_sm_enable_allocate
/use_epv	Generate server side application calls via entry-pt vector
/no_default_epv	Do not generate a default entry-point vector
/prefix client str	Add “str” prefix to client-side entry points
/prefix server str	Add “str” prefix to server-side manager routines
/prefix switch str	Add “str” prefix to switch routine prototypes
/prefix all str	Add “str” prefix to all routines
/win16	Target environment is Microsoft Windows 16-bit (Win 3.x)
/win32	Target environment is Microsoft Windows 32-bit (NT)

A.7 Error and Warning Messages

Switch	Use
/error none	Turn off all error checking options
/error allocation	Check for out of memory errors
/error bounds_check	Check size vs transmission length specification
/error enum	Check enum values to be in allowable range
/error ref	Check ref pointers to be non-null
/error stub_data	Emit additional check for server side stub data validity
/no_warn	Suppress compiler warning messages

A.8 Optimization

Switch	Use
/align {1|2|4|8}	Designate packing level of structures
/pack {1|2|4|8}	Designate packing level of structures
/Zp{1|2|4|8}	Designate packing level of structures
/Oi	Generate fully interpreted stubs
/Oic	Generate fully interpreted stubs for standard interfaces and stubless proxies for object interfaces as of NT 3.51 release
/Oicf	Generate fully interpreted stubs with extensions and stubless proxies for object interfaces as of NT 4.0 release
/Os	Generate inline stubs
/hookole	Generate HookOle debug info for local object interfaces

A.9 Miscellaneous

Switch	Use
@response_file	Accept input from a response file
/?	Display a list of MIDL compiler switches
/confirm	Display options without compiling MIDL source
/help	Display a list of MIDL compiler switches
/nologo	Suppress displaying of the banner lines
/o filename	Redirects output from screen to a file
/W{0|1|2|3|4}	Specify warning level 0-4 (default = 1)
/WX	Report warnings at specified **/W** level as errors

B
Troubleshooting

B.1 RPC Troubleshooting

When you perform a significant number of simultaneous NTLM authentications, the following errors are likely to occur. Several factors affect the number of simultaneous NTLM authentications, however, you are most likely to see these errors when the network is congested or when the RPC application server does not respond to requests in a timely manner. The errors are returned as standard RPC application return values.

Table B–1 provides a description of the suspected cause and possible workarounds.

Table B–1 RPC Errors

Error	Cause/Corrective Actions
RPC_S_CONNECTION_REJECTED	This error is seen by the client application as an exception when using either DECnet Phase IV or DECnet Phase V as a transport and when the server is heavily loaded servicing other DECnet clients. The system returns this error when the client RPC run time binds to a newly created socket and the socket call returns error 61 (connection refused). **Possible solutions:** 1. Raise DECnet resource quotas. 2. Enhance the client RPC program to catch the exception and either retry the RPC or choose a different server.

(continued on next page)

Table B–1 (Cont.) RPC Errors

Error	Cause/Corrective Actions
RPC_S_CONNECTION_TIMED_OUT	This error is seen by the client application as an exception when using TCP or DECnet as a transport and when the server is heavily loaded. The system returns this error when the client RPC run time binds to a newly created socket and the server takes too long to either accept or reject the connection. **Possible solutions:** 1. Configure TCP or DECnet to wait longer before sockets time out. 2. Enhance the RPC client application to call `rpc_mgmt_set_com_timeout()` and instruct the RPC run time to retry when it gets this socket error. 3. Recode the client RPC program to catch the exception and either retry the RPC or choose a different server.
RPC_S_ASSOC_SHUTDOWN	This error is seen by the client application as an exception when using TCP or DECnet as a transport and when the client is heavily loaded (usually when the client is also an RPC server). After an RPC server receives an RPC_BIND packet from a client and the server sends back an RPC_BIND_ACK packet to the client, the server expects to receive a `REQUEST` packet within 12 seconds. If the client does not send the `REQUEST` packet within 12 seconds, the RPC server deletes the association and sends a `SHUTDOWN` packet to the client. The client RPC run time raises an exception to the RPC application. This scenario is likely to occur when the client RPC application is also acting as an RPC server and that RPC server is already heavily loaded. **Possible solutions:** 1. Implement the client RPC program to catch the exception and either retry the RPC or choose a different server.

(continued on next page)

Table B–1 (Cont.) RPC Errors

Error	Cause/Corrective Actions
RPC_S_COMM_FAILURE	This error is seen by the client application as an exception when using DG (UDP) as a transport and when the RPC server is heavily loaded. The RPC client sends a `REQUEST` packet to the server. If the client does not get a `WORKING` packet response from the server within 30 seconds, the client sends a `PING` packet to the server to see if the server is still active and working on the client's request. If the RPC server is under heavy load, the server may not return the `WORKING` packet to the client before the client times out. **Possible solutions:** 1. RPC client application can call `rpc_mgmt_set_com_timeout()` to instruct the RPC run time to wait longer than 30 seconds before timing out. 2. Implement the client RPC program to catch the exception and either retry the RPC or choose a different server.

B.2 Troubleshooting the ACME server

Use the following procedure to troubleshoot problems with the ACME server:

1. Verify that the ACME_SERVER process is running (use the `SHOW SERVER ACME` command) and verify there is a connection between the MSV1_0 ACME agent and the HP Advanced Server for OpenVMS process.
2. If no connection exists, verify that the PWRK$ACME_SERVER logical name contains the SCS node names of systems in the cluster that are running the HP Advanced Server for OpenVMS process.
3. If the PWRK$ACME_SERVER logical name is defined correctly, verify that the HP Advanced Server for OpenVMS process is running on the systems specified (look for the `PWRK$LMSRV` process).
4. If authentications are failing, check the following:
   - Interdomain authentication (EASTOSHKOSK\JOE) requires trust relationships. Use the HP Advanced Server for OpenVMS ADMINISTER ADD TRUST[/TRUSTED] or [/PERMITTED] command to establish the desired trust relationships between two domains.
   - Windows passwords are case sensitive. Be sure you have entered the passwords using the correct case.
   - Windows user either has no OpenVMS hostmap account or maps to an invalid OpenVMS account (the default UAF mapping is PWRK$DEFAULT, which has DISUSER flag set). Use the HP Advanced Server for OpenVMS ADMINISTER ADD HOSTMAP command to map the Windows user name to a valid OpenVMS account.

- Windows user account is invalid, expired, disabled, or has an invalid password. Use the HP Advanced Server for OpenVMS ADMINISTER SHOW USER/FULL command to display the complete user account information. Use the HP Advanced Server for OpenVMS ADMINISTER SHOW ACCOUNT POLICY command to display the domain policy information.
- OpenVMS account does not have EXTAUTH flag set. In AUTHORIZE, use the UAF utility MODIFY *user-name*/FLAG=EXTAUTH command. (You can override this requirement by setting the IGNORE_EXTAUTH bit (bit number 11 [decimal]) in the SECURITY_POLICY system parameter.)
- UAF record flag is set to DISUSER. In AUTHORIZE, use the UAF utility MODIFY *user-name*/FLAG=NODISUSER command.
- UAF record modal restrictions prevent "login" (check local dialup, remote, network, and batch access restrictions). In AUTHORIZE, use the UAF utility MOD *user-name*/LOCAL (or DIALUP, BATCH, NETWORK,REMOTE) keywords; INTERACTIVE sets LOCAL, DIALUP, and REMOTE access restrictions.
- Intrusion subsystem has entered break-in evasion mode because the number of failed logins has exceeded the system threshold (set by SYSGEN parameter LGI_BRK_LIM). Use the SHOW INTRUSION command to view the intrusion database. Use the DELETE/INTRUSION *source* command to remove entries from the database. If the LGI_BRK_ DISUSER is set, the UAF record may be set to DISUSER. Use the OpenVMS AUTHORIZE command to reset the flag.

B.3 Troubleshooting the DCOM$RPCSS Process

The DCOM$RPCSS process must be running to run any COM for OpenVMS applications on your OpenVMS system. The DCOM$STARTUP.COM command file is automatically starts this process. If you have problems running COM for OpenVMS applications, check that this process is running. Use the following command:

```
$ SHOW SYSTEM
```

If the process is initializing, the process name is DCOM$STARTUP-**. If the process is in its normal running state, the process name is be DCOM$RPCSS.

Check the SYS$MANAGER:DCOM$RPCSS.OUT log file for error messages from the DCOM$RPCSS process. The messages can include the following:

- `%ACME-E-PWDEXPIRED, password has expired`

  If the DCOM$RPCSS log file contains this error, do the following:

  1. Run the HP Advanced Server for OpenVMS ADMIN utility and to change the password of the DCOM$RPCSS account. See Section 6.2.1.
  2. Update the COM for OpenVMS Service Control Manager password file. See Section 6.2.1.

B.4 Troubleshooting the Advanced Server for OpenVMS

The Advanced Server for OpenVMS must be running to authenticate users with credentials.

A troubleshooter may wish to enable the audit policy to capture failures for logonoff and system events. For example, on systems running Advanced Server for OpenVMS, issue the command:

```
$ ADMINISTER SET AUDIT POLICY/AUDIT/FAILURE=(LOGONOFF,SYSTEM)
```

To monitor events, issue the commands:

```
$ ADMINISTER SHOW EVENT /FULL /TYPE=SYSTEM
$ ADMINISTER SHOW EVENT /FULL /TYPE=SECURITY
```

For more information, the *Advanced Server for OpenVMS Server Administrator's Guide* provides a chapter on Monitoring Events and Troubleshooting.

Additionally, the system manager may want to check the system operator log, SYSMANAGER:OPERATOR.LOG, to verify that no network errors have occurred.

B.5 Troubleshooting COM for OpenVMS Application Failures

This section describes problems you may encounter when running a COM application.

B.5.1 Access Denied Failures

For information on access denied failures, see Section 5.4.6.

C

Cookbook Examples: Building a Sample Application on OpenVMS

Note

SAMPLE1 and DISPATCH_SAMPLE1 are taken from Dale Rogerson's book, *Inside COM*, published by Microsoft Press.

C.1 COM Example (Sample1)

This example implements a COM client and server in which the component provides two interfaces: IX and IY. The client also queries the component for a third interface, IZ, an interface that the component does not provide.

This example demonstrates connectivity between two OpenVMS systems, between two Windows systems, or between an OpenVMS system and a Windows system.

Note

Before you build the application on OpenVMS, you must run NTA$LOGON and acquire Windows credentials. For more information, see Section 8.2.

C.1.1 OpenVMS Instructions

The following sections describe how to build the application on an OpenVMS system.

C.1.1.1 Building the Application on OpenVMS

Copy files from the DCOM examples directory to your local directory. For example:

```
$ set default mydisk:[mydirectory]
$ copy dcom$examples:[sample1]*.* []
```

To build the application, run the following command procedure:

```
$ @build_sample1
```

If you have MMS, you can use the included description file as follows:

```
$  MMS/DESCRIPTION=BUILD_SAMPLE1.MMS
```

The BUILD file builds and registers both the in-process and out-of-process servers.

C.1.1.2 Registering the Application on OpenVMS

The build procedure automatically registers both DISPCMPNT$SHR.EXE and DISPCMPNT.EXE. To register the components manually, use the following procedure:

- To register the in-process server, use the REGSVR32 utility as follows:

```
$ regsvr32 :== $DCOM$REGSVR32.EXE
$ regsvr32 path-nameDISPCMPNT$SHR.EXE
```

- To unregister the in-process server, use the REGSVR32 utility as follows:

```
$ regsvr32 /u path-nameDISPCMPNT$SHR.EXE
```

- To register the out-of-process server:

```
$ dispcmpnt :== $path-nameDISPCMPNT.EXE
$ dispcmpnt /regserver
```

- To unregister the out-of-process server:

```
$ dispcmpnt /unregserver
```

- To register the Proxy Stub, use the REGSVR32 utility as follows:

```
$ regsvr32 path-namePROXY$SHR.EXE
```

- To unregister the Proxy Stub, use the REGSVR32 utility as follows:

```
$  regsvr32 /u path-namePROXY$SHR.EXE
```

C.1.1.3 Running the Application on OpenVMS as an Out-of-Process Server

To run the sample where the component is an out-of-process server, run DISPCMPNT.EXE. When the system displays the Server: Waiting message from the component, run the client in a separate window or terminal session.

- Window (or terminal session) 1:

```
$ run dispcmpnt
```

- Window (or terminal session) 2:

```
$ client :== $path-nameCLIENT.EXE
For OutProc:
$ client
2
$
```

The client displays the following:

```
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 2 to select the out-of-process server.

C.1.1.4 Running the Application on OpenVMS and Specifying a Remote Server

Run DISPCMPNT.EXE on the system you designate as the remote machine (or server system). The remote system can also be a Windows system. When you receive the Server: Waiting message from the component, run the client on the system you designate as the local machine (or client system). For example:

```
$ client :== $path-nameCLIENT.EXE
$ client remote-system-name
2
$
```

The client displays the following:

```
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 2 to select remote server execution, out-of-process server.

C.1.1.5 Running the Application on OpenVMS as an In-Process Server

To run the sample where the component is an in-process server, run only the client. For example:

```
For InProc:
$ client
1
$
```

The client displays the following:

```
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 1 to select the in-process server.

C.1.2 Windows Instructions

The following sections describe how to build the application on a Windows system.

Note

In order to build Visual C++ applications from a DOS window, you must first set up a number of environment variables. If you did not select the option to have these variables set up automatically when you installed Visual C++, you will need to set them up each time you create a DOS window. To set up these variables, execute the file

```
C:\Program Files\Microsoft Visual Studio\VC98\BIN\VCVARS32.BAT
```

C.1.2.1 Building the Application on Windows

Copy the README-SAMPLE1.TXT file and the following files from the COM examples directory to your Windows system:

```
CLIENT.CXX
CMPT.CXX
CMPT.DEF
MAKE-ONE.
MAKEFILE.BAT
PROXY.DEF
REGISTRY.CXX
REGISTRY.H
SERVER.IDL
```

Build the sample using the MAKEFILE.BAT file. For example:

```
> MAKEFILE
```

The Makefile builds and registers both the in-process and out-of-process servers.

C.1.2.2 Registering the Application on Windows

The build procedure make-one automatically registers DISPCMPNT.DLL, PROXY.DLL, and CMPNT.EXE as follows:

```
regsvr32 -s Dispcmpnt.dll
regsvr32 -s Proxy.dll
Dispcmpnt /RegServer
```

To unregister the application, enter the following:

```
regsvr32 -u Dispcmpnt.dll
regsvr32 -u Proxy.dll
Dispcmpnt /UnRegServer
```

C.1.2.3 Running the Application on Windows

Run CLIENT. Follow the same procedure as described for OpenVMS for running the application as an in-process server (Section C.1.1.5) and out-of-process server Section C.2.1.3).

Use the name of a remote machine (UNC or DNS) as an argument to instantiate the object on the remote machine. For example:

```
>Client hostname        ! point the client at the remote system
2                       ! means outproc invocation
>
```

C.2 Automation Example (Dispatch_Sample1)

This example implements the Automation component server as a dual interface. There are two separate clients: Dclient, which connects to the dual interface through the dispinterface, and Client, which is a COM client implementation that connects through the IUnknown interface (using a v-table).

This example demonstrates connectivity between two OpenVMS systems, between two Windows systems, or between an OpenVMS system and a Windows system.

C.2.1 OpenVMS Instructions

The following sections describe how to build the application on an OpenVMS system.

C.2.1.1 Building the Application on OpenVMS

Copy files from the DCOM examples directory to your local directory. For example:

```
$ set default mydisk:[mydirectory]
$ copy dcom$examples:[dispatch_sample1]*.* []
```

To build the application, run the following command procedure:

```
$ @build_dispatch_sample1
```

If you have MMS, you can use the included description file as follows:

```
$  MMS/DESCRIPTION=BUILD_DISPATCH_SAMPLE1.MMS
```

The BUILD file builds and registers both the in-process and out-of-process servers.

C.2.1.2 Registering the Application on OpenVMS

The build procedure automatically registers both DISPCMPNT$SHR.EXE and DISPCMPNT.EXE. To register the components manually, use the following procedure:

- To register the in-process server, use the REGSVR32 utility as follows:

```
$ regsvr32 :== $DCOM$REGSVR32.EXE
$ regsvr32 path-nameDISPCMPNT$SHR.EXE
```

- To unregister the in-process server, use the REGSVR32 utility as follows:

```
$ regsvr32 /u path-nameDISPCMPNT$SHR.EXE
```

- To register the out-of-process server:

```
$ dispcmpnt :== $path-nameDISPCMPNT.EXE
$ dispcmpnt /regserver
```

- To unregister the out-of-process server:

```
$ dispcmpnt /unregserver
```

C.2.1.3 Running the Application on OpenVMS as an Out-of-process Server

To run the sample where the component is an out-of-process server, run DISPCMPNT.EXE.

When the system displays the Server: Waiting message from the component, run the client in a separate window or terminal session.

- Window (or terminal session) 1:

```
$ run dispcmpnt
```

- Window (or terminal session) 2:
  - For dispatch client:

```
$ run dclient
```

  - For COM client:

```
$ run client
```

The client displays the following:

```
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 2 to select the out-of-process server.

C.2.1.4 Running the Application on OpenVMS and Specifying a Remote Server

Run DISPCMPNT.EXE on the system you designate as the remote machine (or server system). The remote system can also be a Windows system. When you receive the Server: Waiting message from the component, run the client on the system you designate as the local machine (or client system). For example:

To use the COM client, enter the following:

```
$ client :== $path-nameCLIENT.EXE
$ client remote-system-name
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 2 to select remote server execution, out-of-process server.

C.2.1.5 Running the Application on OpenVMS as an In-Process Server

To run the sample where the component is an in-process server, run only the client. For example:

- For dispatch client:

```
$ run dclient
```

- For COM client:

```
$ run client
```

The client displays the following:

```
To which server do you want to connect?
 1) In-Process Server
 2) Out-of-Process Server
 :
```

Enter 1 to select the in-process server.

C.2.2 Windows Instructions

The following sections describe how to build the application on a Windows system.

Note

In order to build Visual C++ applications from a DOS window, you must first set up a number of environment variables. If you did not select the option to have these variables set up automatically when you installed Visual C++, you will need to set them up each time you create a DOS window. To set up these variables, execute the file

```
C:\Program Files\Microsoft Visual Studio\VC98\BIN\VCVARS32.BAT
```

C.2.2.1 Building the Application on Windows

Copy the README-DISPATCH-SAMPLE1.TXT file and the following files from the COM examples directory to your Windows system:

```
CLIENT.CXX
DCLIENT.CXX
DISPCMPNT.CXX
DISPCMPNT.DEF
DISPCMPNT.IDL
MAKE-ONE.
MAKEFILE.BAT
REGISTRY.CXX
REGISTRY.H
```

Build the sample using the MAKEFILE.BAT file. For example:

```
C:> MAKEFILE
```

The Makefile builds and registers both the in-process and out-of-process servers.

C.2.2.2 Registering the Application on Windows

The build procedure make-one automatically registers DISPCMPNT.DLL, PROXY.DLL, and DISPCMPNT.EXE as follows:

```
regsvr32 -s Dispcmpnt.dll
Dispcmpnt /RegServer
```

To unregister the application, enter the following:

```
regsvr32 -u Dispcmpnt.dll
Dispcmpnt /UnRegServer
```

C.2.2.3 Running the Application on Windows

Run DCLIENT or CLIENT. Follow the same procedure as described for OpenVMS for running the application as an in-process server (Section C.2.1.5) and an out-of-process server (Section C.2.1.3).

Use the name of a remote machine (UNC or DNS) as an argument to instantiate the object on the remote machine.

C.3 Cross-Domain Security Example (CLIENTAUTH)

This example shows how you can authenticate a remote client that is not in the server's domain or in a domain that has a trust with the server's domain. The client must pass to this application the credentials (user name, domain and password) of an account on the server's domain that is allowed access and launch permissions. In fact, the client need not be in any domain and can be anywhere on the network. This is demonstrated in Section C.3.1.3.

C.3.1 OpenVMS Instructions

The following sections describe how to build the application on an OpenVMS system.

Note

Not all functionality is present in the underlying WindowsNT infrastructure on OpenVMS. Therefore, you cannot run the client on OpenVMS. This example works when you run the client on Windows and the server on OpenVMS.

Copy files from the DCOM examples directory to your local directory:

```
$ set default mydisk:[mydirectory]
$ copy dcom$examples:[clientauth]*.* []
```

To build the application, run the command procedure:

```
$ @build_clientauth
```

The BUILD file builds and registers both the in-process and out-of-process servers.

C.3.1.1 Registering the Application on OpenVMS

PROXY$SHR.EXE, CLIENTAUTH$SHR.EXE, and CLIENTAUTH.EXE are registered automatically by the build procedure. To register the application manually, use the following procedure:

- To register the in-process server, use the REGSVR32 utility provided:

```
$ regsvr32 :== $DCOM$REGSVR32.EXE
$ regsvr32 <path-name>CLIENTAUTH$SHR.EXE
$ regsvr32 <path-name>PROXY$SHR.EXE
```

- To unregister the in-process server:

```
$ regsvr32 /u <path-name>CLIENTAUTH$SHR.EXE
$ regsvr32 /u <path-name>PROXY$SHR.EXE
```

- To register the out-of-process server:

```
$ clientauth :== $<path-name>CLIENTAUTH.EXE
$ clientauth /regserver
```

- To unregister the out-of-process server:

```
$ clientauth /unregserver
```

C.3.1.2 Running the Application on OpenVMS as an Out-of-Process Server

To run the example when the component is an out-of-process server, run CLIENTAUTH.EXE. When you receive the server waiting message from the component, run the client (in a separate window or terminal session).

- Window (or terminal session) 1:

```
$ run clientauth
```

- Window (or terminal session) 2:

```
$ client :== $<path-name>CLIENT.EXE
For OutProc:
$ client
2
$
```

The client asks whether you want to start an in-process server or an out-of-process server. Specify out-of-process server.

C.3.1.3 Running the Application on OpenVMS and Specifying a Remote Server

Run CLIENTAUTH.EXE on the system you designate as the remote machine, or server system. The remote system can also be a Windows system. When you receive the server waiting message from the component, run the client on the system you designate as the local machine, or client system.

```
$ client :== $<path-name>CLIENT.EXE
$ client <remote-system-name>
2
$ Please enter account to use on remote machine:
$ Username:
$ Domain:
$ Password:
```

The client asks whether you want to start an in-process server or an out-of-process server. For remote server execution, specify out-of-process server. You then are prompted to enter the user name, domain, and password of an account on the remote server. Make sure this account has been granted access and launch permissions to the component (see Section 6.3.2).

C.3.1.4 Running the Application on OpenVMS as an In-Process Server

To run the sample when the component is an in-process server, run only the client:

```
For InProc:
$ client
1
$
```

The client asks whether you want to start an in-process server or an out-of-process server. Specify in-process server.

C.3.2 Windows Instructions

The following sections describe how to build the application on a Windows system.

Note

In order to build Visual C++ applications from a DOS window, you must first set up a number of environment variables. If you did not select the option to have these variables set up automatically when you installed Visual C++, you will need to set them up each time you create a DOS window. To set up these variables, execute the following file:

```
C:\Program Files\Microsoft Visual Studio\VC98\BIN\VCVARS32.BAT
```

C.3.2.1 Building the Application on Windows NT

Copy the file `README-CLIENTAUTH.TXT` and the following files from the DCOM examples directory to your Windows system:

```
CLIENT.CXX
CLIENTAUTH.CXX
CLIENTAUTH.DEF
CLIENTAUTH.IDL
GUIDS.CXX
MAKE-ONE.
MAKEFILE.BAT
PROXY.DEF
REGISTRY.CXX
REGISTRY.H
```

Build the sample using the `MAKEFILE.BAT` file:

```
> MAKEFILE
```

The Makefile builds and registers both the in-process and out-of-process servers.

C.3.2.2 Registering the Application on Windows

`CLIENTAUTH.DLL`, `PROXY.DLL`, and `CLIENTAUTH.EXE` are registered automatically by the build procedure <make-one>:

```
regsvr32 -s clientauth.dll
regsvr32 -s Proxy.dll
clientauth /RegServer
```

To unregister the application:

```
regsvr32 -u clientauth.dll
regsvr32 -u Proxy.dll
clientauth /UnRegServer
```

C.3.2.3 Running the Application on Windows

Run CLIENT. Follow the same procedure as described for running the application on an in-process server and out-of-process server (see Section C.3.1.2 and Section C.3.1.4).

Do not use command line arguments to instantiate the object on the current machine. Instead, use the name of a remote machine (UNC or DNS) as an argument to instantiate the object on the remote machine.

```
(i.e) >Client hostname        ! point the client at the remote system
      2                       ! means outproc invocation
      >
      >Username:
      >Domain:
      >Password:
```

D

Upgrading to COM Version 1.4 for OpenVMS from COM Version 1.0 for OpenVMS

D.1 Upgrading from Earlier Versions of COM for OpenVMS

The following sections describe tasks you must complete when upgrading from a previous version of COM for OpenVMS.

D.1.1 Rebuild Existing COM for OpenVMS Applications

If your COM for OpenVMS applications include references to any of the following APIs, you must recompile the modules that include the references and relink the application:

```
LoadLibraryA
LoadLibraryW
LoadLibraryExW
LoadLibraryExA
GetModuleFileNameA
GetModuleFileNameW
GetModuleHandleW
GetProcAddress
FreeLibrary
```

Some sample COM applications that shipped with COM Version 1.0 for OpenVMS include references to these APIs in the modules REGISTRY and CMPNT. If you built any samples, or if you built your own COM applications based on these samples, you should recompile and relink those applications.

D.1.2 Configuring the Windows Systems

For COM Version 1.0 for OpenVMS (unauthenticated COM) the COM for OpenVMS documentation instructed you to change specific values in your Windows registry to allow unauthenticated COM for OpenVMS to interoperate with Windows. COM Version 1.1-A and higher for OpenVMS support authentication. As a result, you must set or reset the Windows Registry values we asked you to change for COM Version 1.0 for OpenVMS back to their default authenticated settings. To set the Windows Registry values, use the following procedure:

1. Start the Windows Registry editor.
2. Select the following registry key:

   ```
   HKEY_LOCAL_MACHINE\Software\Microsoft\Ole
   ```

3. Delete the following value names and value data:

Value name	Recommended COM V1.0 setting	Default (Authenticated) Value data (COM V1.4 setting)	Registry type
ActivationSecurity	N	Remove	REG_SZ
PersonalClasses	N	Remove	REG_SZ

4. Verify the **Default Authentication Level** and **Default Impersonation Level** and change if necessary. Use the following procedure:

Note

You must have Windows Administrator privileges to view and update these settings.

   a. From the **Start** menu, choose **Run...**
   b. In the Run dialog box, enter dcomcnfg.

      The system displays the *Distributed COM Configuration Properties* sheet.
   c. Click the **Default Properties** tab.
      - The *Default Authentication Level* list box should display **Connect**. If it does not, click the list box arrow and select **Connect** from the list.
      - The *Default Impersonation Level* list box should display **Identity**. If it does not, click the list box arrow and select **Identity** from the list.

5. You must reboot the Windows system for these changes to take effect.

D.1.3 Configuring the OpenVMS System

On OpenVMS systems, you must set or reset the specific OpenVMS Registry values. You can use the Windows Registry editor to edit the OpenVMS Registry, or you can use the REG$CP utility. To set the OpenVMS Registry values, use the following procedure:

1. Select the following OpenVMS Registry key:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Ole
```

2. Delete the ActivationSecurity, PersonalClasses, LegacyAuthenticationLevel, and LegacyImpersonationLevel keys. Use the following commands to delete the keys:

```
$ REG$CP == "$REG$CP"
$ REG$CP LIST VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole
$ REG$CP DELETE VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole ActivationSecurity
$ REG$CP DELETE VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole PersonalClasses
$ REG$CP DELETE VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole LegacyAuthenticationLevel
$ REG$CP DELETE VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole LegacyImpersonationLevel
$ REG$CP LIST VALUE HKEY_LOCAL_MACHINE\Software\Microsoft\Ole
$ REG$CP EXIT
```

D.2 Previously Configured Applications on Windows

If you configured an application to run with COM Version 1.0 for OpenVMS (unauthenticated COM for OpenVMS) on Windows, you might want to reconfigure the Windows settings to take advantage of the most recent version of COM for OpenVMS (authenticated COM for OpenVMS).

Under COM Version 1.0 for OpenVMS after you registered a component, the COM for OpenVMS documentation instructed you to check the security properties on that component to ensure that an unauthenticated user can activate the image. Use the following procedure:

1. From the Windows **Start** menu, choose **Run...**
2. In the Run dialog box, enter `dcomcnfg`.

   The system displays the *Distributed COM Configuration Properties* sheet.
3. Select the object by name from the Applications list, then click the **Properties...** button.

   The system displays the property sheet for the selected object.
4. From the property sheet, click the **Security** tab.

   - For COM Version 1.0 for OpenVMS you had to set the access permissions (Registry value `AccessPermission`) so that user **Everyone** was allowed access (**Allow access**).

     For the most recent version of COM for OpenVMS, you can set custom access permissions (Registry value `AccessPermission`) to a specific user.

     Click *Use custom access permissions* , then click the **Edit** button to display the **Registry Key Permissions** box.
   - For COM Version 1.0 for OpenVMS you had to set the launch permissions (Registry value `LaunchPermission`) so that user **Everyone** was allowed to launch the application server (**Allow launch**).

     For the most recent version of COM for OpenVMS, you can set the custom launch permissions (Registry value `LaunchPermission`) to remove **Everyone**.

     Click *Use custom access permissions* , then click the **Edit** button to display the **Registry Key Permissions** box.
   - For COM Version 1.0 for OpenVMS you had to set the configuration permissions so that user **Everyone** was allowed at least **Read** access to the Registry values.

     For the most recent version of COM for OpenVMS, you can set the custom configuration permissions to remove **Everyone**.

     Click *Use custom access permissions* , then click the **Edit** button to display the **Registry Key Permissions** box.

   Under COM Version 1.0 for OpenVMS after you set security properties, you had to set the identity of the account to run the application.

   For the most recent version of COM for OpenVMS, you can set the identity of the account to option 1 or 2.

   Click the **Identity** tab to display the user account selection. Select *The interactive user* option.

D.2 Previously Configured Applications on Windows

D.2.1 You Must Repopulate the OpenVMS Registry for COM Version 1.4 for OpenVMS

For the most recent version of COM for OpenVMS, you must repopulate the OpenVMS Registry to include security settings. Use the DCOM$SETUP command procedure to display the OpenVMS COM Tools menu, and choose option 3.

D.2.2 Changing Application Security Settings in the OpenVMS Registry

COM Version 1.0 for OpenVMS, which shipped with OpenVMS 7.2, did not support NTLM security. As a result, the OpenVMS account through which you (or the system) registered the COM Version 1.0 for OpenVMS COM application was the owner for any OpenVMS Registry keys created as part of the application registration. For example, using COM Version 1.0 for OpenVMS, if you logged into the SYSTEM account and registered the SAMPLE1 application, all SAMPLE1's OpenVMS Registry keys are owned by SYSTEM.

COM V1.1-A and higher for OpenVMS supports NTLM security. The system now uses the Windows domain account to control access to the OpenVMS Registry keys. As a result of this change, previous security settings might prevent a nonprivileged user from accessing an application's registry keys. This means that a nonprivileged user working on an existing application might not be able to unregister or reregister an application.

To prevent this registration lockout, you must change the permission of the application. You can change the permission from either the Windows system or the OpenVMS system. Use either of the following procedures:

- Changing the permission from a Windows system
  1. From a Windows system, start RegEdt32.
  2. From the **Registry** menu, choose **Select Computer** and connect to the OpenVMS system that contains the OpenVMS Registry.
  3. Select the key associated with the application you want to change.
  4. From the **Security** menu, choose **Permissions...** and grant the user Full Control.
  5. Repeat the last two steps for each registry key associated with the application. For a list of COM application-related registry keys, see Section D.2.2.1.
- Changing the permission from an OpenVMS system
  1. Log into a privileged OpenVMS account.
  2. Unregister the application. Use the DCOM$REGSVR32 utility. See Example 6–5.
  3. Delete all registry keys associated with the application. For a list of COM application-related registry keys, see Section D.2.2.1.
  4. Log into the nonprivileged user account.
  5. Register the application. Use the DCOM$REGSVR32 utility (see Example 6–4), or from the OpenVMS COM Tools menu, choose option 6 (see Section 6.2).

D.2.2.1 COM Application Registry Keys

A COM application can have several registry keys associated with it. You must be sure to change all keys associated with the application. An application usually registers the following keys:

HKEY_CLASSES_ROOT\CLSID\{*guid*} and subkeys
HKEY_CLASSES_ROOT\APPID\{*guid*}
HKEY_CLASSES_ROOT\APPID*filename*
HKEY_CLASSES_ROOT\TYPELIB\{*typelib guid*}
HKEY_CLASSES_ROOT\INTERFACES\{*interface guid(s)*} and subkeys
HKEY_CLASSES_ROOT*name* and subkeys
HKEY_CLASSES_ROOT*version independent name* and subkeys

Note

HKEY_CLASSES_ROOT is an alias for HKEY_LOCAL_MACHINE\SOFTWARE\Classes. If you connect to the OpenVMS Registry from Windows using Regedt32 and you want to edit the HKEY_CLASSES_ROOT key, edit the HKEY_LOCAL_MACHINE\SOFTWARE\Classes key.

E

Running COM for OpenVMS in an Unauthenticated Mode

COM for OpenVMS includes an option that allows you to run the software in an unauthenticated environment in which NTLM support is not utilized. If you enable this option, only OpenVMS security semantics are used to control COM applications' access to resources. This is essentially the same behavior as in COM Version 1.0 for OpenVMS.

For a list of security differences between an unauthenticated implementation and an authenticated implementation of COM for OpenVMS, see Table 1–1.

Note

When you run COM for OpenVMS in unauthenticated mode, detached processes started by `DCOM$RPCSS` to run COM servers run in the context of the OpenVMS `DCOM$GUEST` account. These detached processes have the security attributes of the `DCOM$GUEST` account.

The following sections describe tasks you must complete in order to run COM for OpenVMS in an unauthenticated environment.

E.1 Installing COM for OpenVMS to Run in Unauthenticated Mode

If you are installing COM for OpenVMS for the first time, or if you are upgrading from an earlier version, perform the following steps:

- Follow the installation and upgrade procedures described in Chapter 4. (**Note:** You can skip the steps relating to the installation, configuration, and startup of HP Advanced Server for OpenVMS.)
- Follow the configuration procedures in Section E.2 to configure COM for OpenVMS in unauthenticated mode.

The ACME Server process is started automatically by RPC, but it is not required if you are in unauthenticated mode. To cause the ACME Server process to not start when the system reboots, edit the SYLOGICALS.COM file as follows:

```
$  DEFINE ACME$TO_BE_STARTED FALSE             ! ACME Server
```

E.2 Configuring COM for OpenVMS to Run in Unauthenticated Mode

The following section describes how to configure COM for OpenVMS to run in an unauthenticated environment.

Note

Before you begin configuring COM for OpenVMS for unauthenticated mode, make a note of your current Windows system default values and application settings. This makes returning to authenticated mode easier. (For information about how to convert from unauthenticated mode to authenticated mode, see Section E.5.)

E.2.1 Define the DCOM$UNAUTHENTICATED Logical Systemwide

Define `DCOM$UNAUTHENTICATED` to be Y or YES systemwide. If this logical is undefined or defined as any other value, COM for OpenVMS will run in the usual authenticated mode utilizing NTLM security.

To cause COM for OpenVMS to start automatically in unauthenticated mode when the system boots, edit the SYLOGICALS.COM file and add the following line:

```
$ DEFINE/SYSTEM DCOM$UNAUTHENTICATED YES
```

E.2.2 Populate the OpenVMS Registry

Use option 3 in the `DCOM$SETUP` utility to populate the OpenVMS Registry. (See Section 6.2 for more information.)

Note

If you are upgrading from COM Version 1.1-A for OpenVMS or higher, you do not need to populate the OpenVMS Registry.

E.2.3 Create the DCOM$GUEST Account

Create the OpenVMS `DCOM$GUEST` account using option 7 in the `DCOM$SETUP` utility. (See Section 6.2 for more information.)

E.2.4 Create the DCOM$RPCSS Account

Create the OpenVMS `DCOM$RPCSS` account using option 8 in the `DCOM$SETUP` utility. (See Section 6.2 for more information.)

E.3 Configuring Windows to Interoperate with Unauthenticated COM

For COM objects to interoperate correctly between unauthenticated COM for OpenVMS systems and Windows, perform the steps described in the following sections. This will configure the COM objects to run without security enabled on the Windows system.

E.3.1 Setting the Windows Systemwide Authentication Level

On Windows systems, set the systemwide authentication level using this procedure:

1. Run DCOMCNFG on the Windows system.
2. Select the **Default Properties** tab.
3. Set the **Default Authentication Level** to **None**.

E.3.2 Setting Windows Application Security Properties

After a COM application has been registered, check the security properties for that application to ensure that an unauthenticated user can activate the image.

To do this, perform the following steps:

1. Run DCOMCNFG on the Windows system.
2. Select the application by name.
3. Click the **Properties** button.
4. Click the **Security** tab.

Set the access permissions (registry value AccessPermission) so that user **Everyone** is allowed access (**Allow** access).

Set the launch permissions (registry value LaunchPermission) so that user **Everyone** is allowed access (**Allow** access).

Set the configuration permissions so that user **Everyone** is allowed at least **Read** access to the Registry values.

E.3.3 Setting the Windows Application Security Identity

After you set security permissions, you must set the identity of the account to run the application. To do this, click the **Identity** tab, and select **The interactive user**.

E.4 Expected Failures from CLIENTAUTH Sample Program

While you are running COM for OpenVMS in unauthenticated mode, the Cross-Domain Security example (CLIENTAUTH) does not work because it requires NTLM authentication to be enabled.

E.5 Converting from Unauthenticated Mode to Authenticated Mode

If you performed the steps in this appendix to run COM for OpenVMS in unauthenticated mode and you want to return to authenticated mode, perform the following steps.

1. Log in to the SYSTEM account.
2. Stop the COM server. Use option 5 in the DCOM$SETUP utility. (See Section 6.2 for more information.)
3. Edit SYLOGICALS.COM with the following changes:
   - Undefine the DCOM$UNAUTHENTICATED logical by entering:

```
$ DEFINE/SYSTEM DCOM$UNAUTHENTICATED NO
```

- Comment the following line, as shown:

```
$! DEFINE ACME$TO_BE_STARTED FALSE      ! ACME Server
```

4. Enter the following command:

```
$ DEFINE/SYSTEM DCOM$UNAUTHENTICATED NO
```

5. Install, configure, and start HP Advanced Server for OpenVMS, if it is not already present.
6. Repopulate the OpenVMS Registry.

   To do this, use option 3 in the DCOM$SETUP utility. (See Section 6.2 for more information.)
7. Add the DCOM$RPCSS account to include the HP Advanced Server for OpenVMS account and hostmap. Use option 8 in the DCOM$SETUP utility. (See Section 6.2 for more information.)
8. Reset your Windows system default values and application settings to the values that were set before you followed the procedure in Section E.3.
9. Start the COM server. Use option 4 in the DCOM$SETUP utility. (See Section 6.2 for more information.)
10. Update or add Windows domain accounts. (See Section 5.1 for more information.)

F

Lists of Differences, APIs, and Interfaces

This appendix contains a list of implementation differences between COM for OpenVMS and Microsoft COM as well as a list of APIs and interfaces provided in this release of COM for OpenVMS.

F.1 Differences between COM for OpenVMS and Microsoft COM

The following sections list important implementation differences between COM for OpenVMS and Microsoft's COM.

F.1.1 Service Control Manager (SCM)

OpenVMS does not provide an equivalent to the Windows Service Control Manager. As a result, applications that depend on Server services (such as stop, start, pause, and resume) rely on the OpenVMS features that provide similar functionality (if the features are available).

For example, you would use the OpenVMS site-specific startup and shutdown command procedures to implement automatic starting of services at system startup and automatic shutdown of services at system shutdown. Service APIs such as `RegisterServiceCtrlHandler`, `ChangeServiceConfig`, and so on, are not provided on OpenVMS.

F.1.2 Server Application Stack Size

In COM for OpenVMS, server application functions run in the context of server threads. As a result, server functions have a limited stack space of 48 KB. If you require additional space for local variables or structures, you should allocate dynamic memory for local variables or structures.

F.1.3 Use of the "char" Datatype

OpenVMS and Windows translate the IDL base data type "char" differently.

OpenVMS translates the data type as MIDL_CHAR, which is defined to be CHAR, and further defined to be "char." The OpenVMS compiler by default takes this to be equivalent to "unsigned char;" in most cases they can be used interchangeably. The two are *not* the same—C++ treats them as different data types you specify them in class member definitions.

Windows translates the data type directly as "unsigned char." This causes conflicts with Visual C++, which treats the "char" datatype as equivalent to "signed char." As in OpenVMS, "char" is not the same as "signed char" in class member definitions.

There are two workarounds to this mismatch:

- Use the data type "CHAR" instead of "char" in the IDL file and all member definitions. This is the most portable solution; you can expect this to work on other systems (such as UNIX) as well.

- Conditionally compile the method definitions so that OpenVMS sees the object methods defined as "char" and Windows sees the methods defined as "unsigned char."

F.1.4 MIDL Compiler Version

The MIDL compiler supplied with COM for OpenVMS is based on Microsoft's MIDL compiler V3.01.76.

F.1.4.1 The OpenVMS MIDL Compiler

The OpenVMS MIDL compiler is identical to the Microsoft Interface Definition Language (MIDL) compiler V3.01.76 except for the following:

1. The Microsoft MIDL implementation supports several optimization levels. The OpenVMS MIDL implementation supports only `-Oicf`. Do not use any other optimization level.
2. The `/cpp_cmd` and `/cpp_opt` switches are not fully functional in the OpenVMS MIDL implementation.
3. On a Windows system, Microsoft MIDL commands, switches, and qualifiers are case sensitive. The OpenVMS MIDL compiler is not case sensitive; all commands, switches, and qualifiers passed to the OpenVMS MIDL compiler are lowercase. As a result, the Microsoft MIDL switches `/I` and `/i` are equivalent on OpenVMS.
4. MIDL-generated files are platform specific.

   You must run MIDL on both platforms. The MIDL output files generated on one platform (OpenVMS or Windows) cannot be copied and used on the other platform.
5. MIDL `-w` switch

   The Microsoft MIDL compiler allows you to specify either `-w` or `-warn` to limit the level of warnings generated by the compiler. The OpenVMS MIDL compiler supports only the `-w` switch.

F.1.5 Using DCOM$CNFG to Change Application Configuration Permission

Use the Application Security Submenu options 5 and 6 to change the OpenVMS Registry key permissions of some keys associated with an application. Option 5 and 6 affect the security settings of the following keys:

`HKEY_CLASSES_ROOT\APPID\{`*guid*`}`
`HKEY_CLASSES_ROOT\CLSID\{`*guid*`}` and subkeys

On Windows systems, the security settings of the subkeys under `HKEY_CLASSES_ROOT\CLSID\{`*guid*`}` are changed *only if the existing security settings match* the original settings of `HKEY_CLASSES_ROOT\APPID\{`*guid*`}`.

On OpenVMS systems, the settings of the subkeys are changed *even if the existing settings do not match* the original settings of `HKEY_CLASSES_ROOT\APPID\{`*guid*`}`.

Options 5 and 6 do not change the settings of all keys associated with an application. For example, options 5 and 6 do not affect the following keys:

`HKEY_CLASSES_ROOT\APPID\`*filename*
`HKEY_CLASSES_ROOT\TYPELIB\{`*typelib guid*`}`
`HKEY_CLASSES_ROOT\INTERFACES\{`*interface guid(s)*`}` and subkeys.
`HKEY_CLASSES_ROOT\`*name* and subkeys
`HKEY_CLASSES_ROOT\`*version independent name* and subkeys

To change the security settings of these keys, use the following procedure:

1. From a Windows system, start RegEdt32.
2. From the **Registry** menu, choose **Select Computer** and connect to the OpenVMS system that contains the OpenVMS Registry.
3. Select the key associated with the application you want to change.
4. From the **Security** menu, choose **Permissions...** and grant the user Full Control.
5. Repeat the last two steps for each registry key associated with the application (see the list of keys described earlier in this section).

Note

`HKEY_CLASSES_ROOT` is an alias for `HKEY_LOCAL_MACHINE\SOFTWARE\Classes`. If you connect to the OpenVMS Registry from Windows using Regedt32 and you want to edit the `HKEY_CLASSES_ROOT` key, edit the `HKEY_LOCAL_MACHINE\SOFTWARE\Classes` key.

F.2 APIs

APIs that require security support are not supported in COM Version 1.0 for OpenVMS.

The APIs supported in this release are as follows:

```
BindMoniker
BstrFromVector
CLSIDFromProgID
CLSIDFromString
CoAddRefServerProcess
CoCopyProxy
CoCreateErrorInfo
CoCreateFreeThreadedMarshaler
CoCreateGuid
CoCreateInstance
CoCreateInstanceEx
CoDisconnectObject
CoDosDateTimeToFileTime
CoFileTimeNow
CoFileTimeToDosDateTime
CoFreeAllLibraries
CoFreeLibrary
CoFreeUnusedLibraries
CoGetCallContext
CoGetClassObject
CoGetCurrentProcess
CoGetErrorInfo
CoGetInstanceFromFile
CoGetInstanceFromIStorage
CoGetInterfaceAndReleaseStream
CoGetMalloc
CoGetMarshalSizeMax
CoGetPSClsid
CoGetStandardMarshal
CoGetTreatAsClass
```

```
CoImpersonateClient
CoInitialize
CoInitializeEx
CoInitializeSecurity
CoIsHandlerConnected
CoLoadLibrary
CoLockObjectExternal
CoMarshalInterface
CoQueryAuthenticationServices
CoQueryClientBlanket
CoQueryProxyBlanket
CoRegisterChannelHook
CoRegisterClassObject
CoRegisterMallocSpy
CoRegisterMessageFilter
CoRegisterPSClsid
CoReleaseMarshalData
CoReleaseServerProcess
CoResumeClassObjects
CoRevertToSelf
CoRevokeClassObject
CoRevokeMallocSpy
CoSetErrorInfo
CoSetProxyBlanket
CoSuspendClassObjects
CoTaskMemAlloc
CoTaskMemFree
CoTaskMemRealloc
CoTreatAsClass
CoUninitialize
CoUnmarshalInterface
CreateAntiMoniker
CreateBindCtx
CreateClassMoniker
CreateDataAdviseHolder
CreateDispTypeInfo
CreateErrorInfo
CreateGenericComposite
CreateILockBytesOnHGlobal
CreateItemMoniker
CreatePointerMoniker
CreateStdDispatch
CreateStreamOnHGlobal
CreateTypeLib
DispGetIDsOfNames
DispGetParam
DispInvoke
DllCanUnloadNow
DllGetClassObject
DllGetClassObject
DllMain
DllRegisterServer
DllUnregisterServer
DosDateTimeToVariantTime
FreePropVariantArray
GetActiveObject
GetAltMonthNames
GetClassFile
GetConvertStg
GetErrorInfo
GetHGlobalFromILockBytes
GetHGlobalFromStream
GetRunningObjectTable
IIDFromString
IsEqualCLSID
```

```
IsEqualGUID
IsEqualIID
IsValidIid
IsValidInterface
IsValidPtrIn
IsValidPtrOut
LHashValOfName
LHashValOfNameSys
LoadRegTypeLib
LoadTypeLibEx
MkParseDisplayName
MonikerCommonPrefixWith
MonikerRelativePathTo
ProgIDFromCLSID
PropStgNameToFmtId
PropVariantClear
PropVariantCopy
QueryPathOfRegTypeLib
ReadClassStg
ReadClassStm
ReadFmtUserTypeStg
RegisterActiveObject
RegisterTypeLib
ReleaseStgMedium
RevokeActiveObject
SafeArrayAccessData
SafeArrayAllocData
SafeArrayAllocDescriptor
SafeArrayCopy
SafeArrayCopyData
SafeArrayCreate
SafeArrayCreateVector
SafeArrayDestroy
SafeArrayDestroyData
SafeArrayDestroyDescriptor
SafeArrayGetDim
SafeArrayGetElement
SafeArrayGetElemsize
SafeArrayGetLBound
SafeArrayGetUBound
SafeArrayLock
SafeArrayPtrOfIndex
SafeArrayPutElement
SafeArrayRedim
SafeArrayUnaccessData
SafeArrayUnlock
SetConvertStg
SetErrorInfo
StgCreateDocfile
StgCreateDocfileOnILockBytes
StgCreatePropSetStg
StgCreatePropStg
StgIsStorageFile
StgIsStorageILockBytes
StgOpenPropStg
StgOpenStorage
StgOpenStorageOnILockBytes
StgSetTimes
StringFromCLSID
StringFromGUID2
StringFromIID
SysAllocString
SysAllocStringByteLen
SysAllocStringLen
SysFreeString
```

```
SysReAllocString
SysReAllocStringLen
SysStringByteLen
SysStringLen
SystemTimeToVariantTime
UnRegisterTypeLib
VarDateFromUdate
VarNumFromParseNum
VarParseNumFromStr
VarUdateFromDate
VariantChangeType
VariantChangeTypeEx
VariantClear
VariantCopy
VariantCopyInd
VariantInit
VariantTimeToDosDateTime
VariantTimeToSystemTime
VectorFromBstr
WriteClassStg
WriteClassStm
WriteFmtUserTypeStg
```

F.3 Interfaces

The interfaces supported in this release are as follows:

```
IAdviseSink
IBindCtx
IClassActivator
IClassFactory
IConnectionPoint
IConnectionPointContainer
ICreateErrorInfo
ICreateTypeInfo
ICreateTypeLib
IDataAdviseHolder
IDataObject
IDispatch
IEnumCallBack
IEnumConnectionPoints
IEnumConnections
IEnumFORMATETC
IEnumMoniker
IEnumOLEVerb
IEnumSTATDATA
IEnumSTATPROPSETSTG
IEnumSTATSTG
IEnumString
IEnumUnknown
IEnumVariant
IErrorInfo
IExternalConnection
ILockBytes
IMalloc
IMallocSpy
IMarshal
IMessageFilter
IMoniker
IMultiQI
IParseDisplayName
IPersist
IPersistFile
IPersistStorage
IPersistStream
```

```
IPropertySetStorage
IPropertyStorage
IRootStorage
IRunnableObject
IRunningObjectTable
IStdMarshalInfo
IStorage
IStream
ISupportErrorInfo
ITypeComp
ITypeInfo
ITypeInfo2
ITypeLib
ITypeLib2
IUnknown
```

G

List of Files Installed by COM for OpenVMS

G.1 Files Installed by COM for OpenVMS

The following files are installed as part of the COM for OpenVMS installation process:

```
[000000]DEC-AXPVMS-DCOM-V0104--1.PCSI$TLB
[DCOM$LIBRARY]ATLBASE.H
[DCOM$LIBRARY]ATLCOM.H
[DCOM$LIBRARY]ATLCONV.CPP
[DCOM$LIBRARY]ATLCONV.H
[DCOM$LIBRARY]ATLDEF.H
[DCOM$LIBRARY]ATLIFACE.H
[DCOM$LIBRARY]ATLIFACE.IDL
[DCOM$LIBRARY]ATLIMPL.CPP
[DCOM$LIBRARY]ATLMAIN.CXX
[DCOM$LIBRARY]CDERR.H
[DCOM$LIBRARY]CGUID.H
[DCOM$LIBRARY]COGUID.H
[DCOM$LIBRARY]COMCAT.H
[DCOM$LIBRARY]COMCAT.IDL
[DCOM$LIBRARY]COMMDLG.H
[DCOM$LIBRARY]CONIO.H
[DCOM$LIBRARY]CRTDBG.H
[DCOM$LIBRARY]DCOM$GUIDGEN.CLD
[DCOM$LIBRARY]DCOM$REGDATA.REG
[DCOM$LIBRARY]DCOM$RUNSHRLIB.CLD
[DCOM$LIBRARY]DCOM.OPT
[DCOM$LIBRARY]DDE.H
[DCOM$LIBRARY]DDEML.H
[DCOM$LIBRARY]DLGS.H
[DCOM$LIBRARY]EXCPT.H
[DCOM$LIBRARY]IMM.H
[DCOM$LIBRARY]INITGUID.H
[DCOM$LIBRARY]LZEXPAND.H
[DCOM$LIBRARY]MCX.H
[DCOM$LIBRARY]MIDLES.H
[DCOM$LIBRARY]MIDL_STUB_TYPES.H
[DCOM$LIBRARY]MMSYSTEM.H
[DCOM$LIBRARY]NB30.H
[DCOM$LIBRARY]NTA_MESSAGE.H
[DCOM$LIBRARY]OAIDL.ACF
[DCOM$LIBRARY]OAIDL.H
[DCOM$LIBRARY]OAIDL.IDL
[DCOM$LIBRARY]OBJBASE.H
[DCOM$LIBRARY]OBJIDL.H
[DCOM$LIBRARY]OBJIDL.IDL
[DCOM$LIBRARY]OCIDL.ACF
[DCOM$LIBRARY]OCIDL.H
[DCOM$LIBRARY]OCIDL.IDL
[DCOM$LIBRARY]OLE2.H
[DCOM$LIBRARY]OLEAUTO.H
[DCOM$LIBRARY]OLECTL.H
[DCOM$LIBRARY]OLEIDL.H
```

G.1 Files Installed by COM for OpenVMS

```
[DCOM$LIBRARY]OLEIDL.IDL
[DCOM$LIBRARY]POPPACK.H
[DCOM$LIBRARY]PRSHT.H
[DCOM$LIBRARY]PSHPACK1.H
[DCOM$LIBRARY]PSHPACK2.H
[DCOM$LIBRARY]PSHPACK4.H
[DCOM$LIBRARY]PSHPACK8.H
[DCOM$LIBRARY]PTHREAD.H
[DCOM$LIBRARY]PTHREAD_EXCEPTION.H
[DCOM$LIBRARY]RPC.H
[DCOM$LIBRARY]RPCDCE.H
[DCOM$LIBRARY]RPCDCEP.H
[DCOM$LIBRARY]RPCNDR.H
[DCOM$LIBRARY]RPCNSI.H
[DCOM$LIBRARY]RPCNSIP.H
[DCOM$LIBRARY]RPCNTERR.H
[DCOM$LIBRARY]RPCPROXY.H
[DCOM$LIBRARY]SERVPROV.H
[DCOM$LIBRARY]SERVPROV.IDL
[DCOM$LIBRARY]SHELLAPI.H
[DCOM$LIBRARY]SHLWAPI.H
[DCOM$LIBRARY]STATREG.CPP
[DCOM$LIBRARY]STATREG.H
[DCOM$LIBRARY]STDOLE2.TLB
[DCOM$LIBRARY]STDOLE32.TLB
[DCOM$LIBRARY]TCHAR.H
[DCOM$LIBRARY]UCS2_DEFINES.HXX
[DCOM$LIBRARY]UNKNWN.H
[DCOM$LIBRARY]UNKNWN.IDL
[DCOM$LIBRARY]URLMON.H
[DCOM$LIBRARY]URLMON.IDL
[DCOM$LIBRARY]UUID.OLB
[DCOM$LIBRARY]VMS_ATL.H
[DCOM$LIBRARY]VMS_DCOM.H
[DCOM$LIBRARY]VMS_IOCTL.H
[DCOM$LIBRARY]WCHAR.H
[DCOM$LIBRARY]WINBASE.H
[DCOM$LIBRARY]WINCON.H
[DCOM$LIBRARY]WINDEF.H
[DCOM$LIBRARY]WINDOWS.H
[DCOM$LIBRARY]WINERROR.H
[DCOM$LIBRARY]WINGDI.H
[DCOM$LIBRARY]WINNETWK.H
[DCOM$LIBRARY]WINNLS.H
[DCOM$LIBRARY]WINNT.H
[DCOM$LIBRARY]WINPERF.H
[DCOM$LIBRARY]WINREG.H
[DCOM$LIBRARY]WINSOCK.H
[DCOM$LIBRARY]WINSPOOL.H
[DCOM$LIBRARY]WINSVC.H
[DCOM$LIBRARY]WINUSER.H
[DCOM$LIBRARY]WINVER.H
[DCOM$LIBRARY]WTYPES.H
[DCOM$LIBRARY]WTYPES.IDL
[SYS$STARTUP]DCOM$RPCSS.COM
[SYS$STARTUP]DCOM$SHUTDOWN.COM
[SYS$STARTUP]DCOM$STARTUP.COM
[SYSEXE]DCOM$CNFG.EXE
[SYSEXE]DCOM$COMREGEDT.EXE
[SYSEXE]DCOM$DLLHOST.EXE
[SYSEXE]DCOM$GUIDGEN.EXE
[SYSEXE]DCOM$MIDL.EXE
[SYSEXE]DCOM$REGSVR32.EXE
[SYSEXE]DCOM$RPCSS.EXE
[SYSEXE]DCOM$RUNSHRLIB.EXE
```

```
[SYSEXE]DCOM$SCLIENT.EXE
[SYSEXE]DCOM$SSERVER.EXE
[SYSEXE]DCOM$SSERVER_REG.COM
[SYSEXE]DCOM$TOOL.EXE
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]BUILD_CLIENTAUTH.COM
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]CLIENTAUTH$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]CLIENTAUTH.CXX
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]CLIENTAUTH.DEF
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]CLIENTAUTH.IDL
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]PROXY$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]PROXY.DEF
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]README-CLIENTAUTH.TXT
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]REGISTRY.CXX
[SYSHLP.EXAMPLES.DCOM.CLIENTAUTH]REGISTRY.H
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]BUILD_DISPATCH_SAMPLE1.COM
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]BUILD_DISPATCH_SAMPLE1.MMS
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]DCLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]DISPCMPNT$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]DISPCMPNT.CXX
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]DISPCMPNT.DEF
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]DISPCMPNT.IDL
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]README-DISPATCH-SAMPLE1.TXT
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]REGISTRY.CXX
[SYSHLP.EXAMPLES.DCOM.DISPATCH_SAMPLE1]REGISTRY.H
[SYSHLP.EXAMPLES.DCOM.EVENTS]BUILD_EVENTS_SAMPLE.COM
[SYSHLP.EXAMPLES.DCOM.EVENTS]EVENTS_SAMPLE.C
[SYSHLP.EXAMPLES.DCOM.EVENTS]EVENTS_SAMPLE.H
[SYSHLP.EXAMPLES.DCOM.EVENTS]NTA_WIN32.C
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]BUILD_SAMPLE1.COM
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]BUILD_SAMPLE1.MMS
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]CMPNT$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]CMPNT.CXX
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]CMPNT.DEF
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]PROXY$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]PROXY.DEF
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]README-SAMPLE1.TXT
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]REGISTRY.CXX
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]REGISTRY.H
[SYSHLP.EXAMPLES.DCOM.SAMPLE1]SERVER.IDL
[SYSHLP.EXAMPLES.DCOM.SIMPLE]BUILD_SIMPLE.COM
[SYSHLP.EXAMPLES.DCOM.SIMPLE]INSTALL.BAT
[SYSHLP.EXAMPLES.DCOM.SIMPLE]MAKEFILE.
[SYSHLP.EXAMPLES.DCOM.SIMPLE]README-SIMPLE.TXT
[SYSHLP.EXAMPLES.DCOM.SIMPLE]REGISTER_SIMPLE.COM
[SYSHLP.EXAMPLES.DCOM.SIMPLE]SCLIENT.CPP
[SYSHLP.EXAMPLES.DCOM.SIMPLE]SSERVER.CPP
[SYSHLP.EXAMPLES.DCOM.SIMPLE]SSERVER.REG
[SYSHLP.EXAMPLES.DCOM.SURROGATE]BUILD_SURROGATE.COM
[SYSHLP.EXAMPLES.DCOM.SURROGATE]BUILD_SURROGATE.MMS
[SYSHLP.EXAMPLES.DCOM.SURROGATE]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.SURROGATE]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.SURROGATE]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.SURROGATE]PROXY$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.SURROGATE]PROXY.DEF
[SYSHLP.EXAMPLES.DCOM.SURROGATE]README-SURROGATE.TXT
[SYSHLP.EXAMPLES.DCOM.SURROGATE]REGISTRY.H
```

```
[SYSHLP.EXAMPLES.DCOM.SURROGATE]REG_SURROGATE.CXX
[SYSHLP.EXAMPLES.DCOM.SURROGATE]SURROGATE$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.SURROGATE]SURROGATE.CXX
[SYSHLP.EXAMPLES.DCOM.SURROGATE]SURROGATE.DEF
[SYSHLP.EXAMPLES.DCOM.SURROGATE]SURROGATE.IDL
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]BUILD_TESTATL_INPROC.COM
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]BUILD_TESTATL_INPROC.MMS
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATH101$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATH101.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATH101.IDL
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATH101PS$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATHFORMULAS.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATHFORMULAS.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]MATHFORMULAS.RGS
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]README-TESTATL_INPROC.TXT
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]RESOURCE.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]STDAFX.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_INPROC]STDAFX.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]BUILD_TESTATL_OUTPROC.COM
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]BUILD_TESTATL_OUTPROC.MMS
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]CLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]INSIDEDCOM.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]INSIDEDCOM.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]INSIDEDCOM.RGS
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]README-TESTATL_OUTPROC.TXT
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]RESOURCE.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]STDAFX.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]STDAFX.H
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]TESTATL.CXX
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]TESTATL.IDL
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]TESTATL.RGS
[SYSHLP.EXAMPLES.DCOM.TESTATL_OUTPROC]TESTATLPS$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]BUILD_INVENTORYCONTROLLER.COM
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]DEFAULT.ASP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]GLOBAL.ASA
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTR$SHR.OPT
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTR.CXX
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTR.DEF
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTR.IDL
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRCLIENT.CXX
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRPROX.CPP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRPROX.H
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRPROXY.CPP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRPROXY.DLL
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]INVCTRPROXY.IDL
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]ORDER.ASP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]README-WEBSAMPLE.TXT
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]REGISTRY.CXX
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]REGISTRY.H
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]RESET.ASP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]STATS.ASP
[SYSHLP.EXAMPLES.DCOM.WEBSAMPLE]STATUS.ASP
[SYSHLP.EXAMPLES.DCOM.WRAPPER]BUILD_WRAPPER.COM
[SYSHLP.EXAMPLES.DCOM.WRAPPER]MAKE-ONE.
[SYSHLP.EXAMPLES.DCOM.WRAPPER]MAKEFILE.BAT
[SYSHLP.EXAMPLES.DCOM.WRAPPER]README.TXT
[SYSHLP.EXAMPLES.DCOM.WRAPPER]REGISTRY.CXX
[SYSHLP.EXAMPLES.DCOM.WRAPPER]REGISTRY.H
[SYSHLP.EXAMPLES.DCOM.WRAPPER]TEST.COM
[SYSHLP.EXAMPLES.DCOM.WRAPPER]VBCLIENT.FRM
[SYSHLP.EXAMPLES.DCOM.WRAPPER]VBCLIENT.VBP
[SYSHLP.EXAMPLES.DCOM.WRAPPER]WR$SHR.OPT
```

```
[SYSHLP.EXAMPLES.DCOM.WRAPPER]WRAPPER.CXX
[SYSHLP.EXAMPLES.DCOM.WRAPPER]WRAPPER.DEF
[SYSHLP.EXAMPLES.DCOM.WRAPPER]WRAPPER.IDL
[SYSHLP.EXAMPLES.DCOM.WRAPPER]WRAPPERCLIENT.CXX
[SYSHLP]COM_REG_EVENTS_DEV_GD.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD.PDF
[SYSHLP]COM_REG_EVENTS_DEV_GD.PS
[SYSHLP]COM_REG_EVENTS_DEV_GD_001.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_002.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_003.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_004.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_005.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_006.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_CONTENTS.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_CONTENTS_001.HTML
[SYSHLP]COM_REG_EVENTS_DEV_GD_INDEX.HTML
[SYSHLP]HP_LOGO.GIF
[SYSHLP]VM-0126A.GIF
[SYSHLP]VM-0224A.GIF
[SYSHLP]VM-0225A.GIF
[SYSHLP]VM-0226A.GIF
[SYSHLP]VM-0227A.GIF
[SYSHLP]VM-0228A.GIF
[SYSHLP]VM-0283A.GIF
[SYSHLP]VM-0331A.GIF
[SYSHLP]VM-8782A.GIF
[SYSHLP]ZK-8782A.GIF
[SYSLIB]DCOM$MIDL_SHR.EXE
[SYSLIB]DCOM$NT_WRAPPERS_SHR.EXE
[SYSLIB]DCOM$OLE32_SHR.EXE
[SYSLIB]DCOM$OLEAUT32_SHR.EXE
[SYSLIB]DCOM$RPCRT4_SHR.EXE
[SYSLIB]DCOM$WIN32_SHR.EXE
[SYSMGR]DCOM$CREATE_ACCOUNT.COM
[SYSMGR]DCOM$REGISTRY_KEYS.COM
[SYSMGR]DCOM$SETUP.COM
[SYSMSG]DCOM$GUIDGEN_MSG.EXE
[SYSMSG]NTADISPMSG.EXE
[SYSMSG]NTAITFMSG.EXE
[SYSMSG]NTARPCMSG.EXE
[SYSMSG]NTAWINMSG.EXE
[SYSMSG]NTAWNDWSMSG.EXE
[000000]DEC-AXPVMS-DCOM-V0104--1.PCSI$DESCRIPTION
```

H

Glossary

class (registry class)

Registry element attribute that allows you to store additional descriptive information with a registry key or subkey.

encapsulation

The process of updating or extending the life of existing application code by leaving most of the code and its functionality intact, while including new or updated code (usually in a different programming language) at key entry points.

For example, you might add a Windows graphical interface to a character-cell application by writing some Visual Basic code that collects information from a Windows client, then formats and submits the data to the existing character cell application as if the data had come from the character cell interface.

hive

A discrete set of keys, subkeys, and value entries contained in the registry.

in-process server

An application that is located on the same system as the requesting client. On Windows systems, in-process servers are usually implemented as DLLs. On OpenVMS systems, in-process servers are usually implemented as shareable images.

key (registry key)

Registry element that contains information specific to the computer, system, or user.

out-of-process server

An application that is located on a different system than the requesting client. On Windows systems, out-of-process servers are usually implemented as .EXE files.

registry

A hierarchical database consisting of one or more files that stores configuration information about system hardware and software.

subkey (registry subkey)

Registry element that is a child of a registry key. A registry key can have zero or more subkeys.

value (registry value)

Registry element that is the entry or value for a registry key or subkey.

wrapper
See **encapsulation**.

I

Acronyms

ACM
Authentication and Credential Management Authority

ACME
Authentication and Credential Management Extension

API
Application Program Interface

ATL
Active Template Library

COM
Component Object Model

CLSID
Class ID

DCOM
Distributed Component Object Model

DLL
Dynamic Link Library

FMS
Forms Management System

GUI
Graphical User Interface

GUID
Globally Unique Identifier

MIDL
Microsoft Interface Definition Language

OO
Object oriented

RPC
Remote Procedure Call

SAM
Security Account Manager

SID
Security Identifier

SMG
Screen Management Facility

SSPI
Security Support Provider Interface

UI
User Interface

UIC
User Identification Code

Index

D

E

G

H

I

R

S

T

U

V

W