BASEstar™ Classic DAS for OMNI™ Software

Installation and User's Guide

Order Number: AA-R21CB-TE

April 2000

This manual describes how to install and use the DAS for OMNI Software for BASEstar Classic on OpenVMS.

Revision/Update Information:	This is a revised document.
Operating System and Version:	OpenVMS/Alpha Version 6.1
Operating System and Version:	OpenVMS/VAX Version 5.5-2
Interface Software and Version:	BASEstar Classic Version 3.4
Network Software and Version:	OMNI/Alpha Version 2.1
Network Software and Version:	OMNI/VAX Version 2.0A
Software Version:	BASEstar Classic DAS for OMNI Software, Version 3.4A

Compaq Computer Corporation
Houston, Texas

April 2000

© 1990 Compaq Computer Corporation.

COMPAQ, DECnet, VMS, and VAX Registered in U.S. Patent and Trademark Office.

BASEstar, OMNI, OSAP and OpenVMS are trademarks of Compaq Information Technologies Group, L.P. in the United States and/or other countries.

S7, S5, SINEC-H1, SINEC-AP and SIMATIC are trademarks of Siemens AG.

All other products mentioned herein may be trademarks or registered trademarks of their respective companies.

Confidential computer software. Valid license from Compaq or authorized sublicensor required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentaion, and Technical data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Compaq shall not be liable for technical or editorial errors or omissions contained herein. The information in this document is subject to change without notice.

This document is available on CD–ROM.

This document was prepared using DECdocument, Version 3.3.

Contents

Preface vii

1 Overview

1.1 Description 1–1
1.2 Device Communications 1–1
1.3 Supported Functions 1–2
1.3.1 Supported MMS Functions 1–3
1.3.2 Supported SINEC-AP and SINEC-H1 Functions 1–4
1.3.3 Supported SINEC-S7 Functions 1–5
1.4 Non-supported Functions 1–6

2 Installing and Configuring the DAS

2.1 Installation Requirements 2–1
2.1.1 Hardware Requirements 2–1
2.1.2 Software Requirements 2–1
2.1.3 Additional Software Requirements 2–2
2.1.4 Disk Space Requirements 2–2
2.2 Installation Procedure 2–3
2.2.1 Files Created During Installation 2–8
2.2.2 Installation Messages 2–8
2.3 Post Installation Tasks 2–10
2.3.1 Configuring layered products 2–10
2.3.2 Editing the Configuration File 2–10
2.3.2.1 Editing Type Records 2–11
2.3.2.2 Editing Path Records 2–12
2.3.2.3 Editing Device Records 2–13
2.3.2.4 NCL Configuration 2–16
2.3.3 Tuning ILAN$DEVSRV 2–18
2.3.4 Defining OMNI Constants 2–19
2.3.5 Setting the SPT Block Parameter 2–19
2.3.6 Setting Up Shop Floor Equipment 2–20
2.4 Tracing OSI Messages 2–20
2.5 Failures During Product Use 2–20

3 Using the DAS

3.1 Accessing OMNI Functions 3–1
3.2 Supported Functions 3–1
3.2.1 Connecting To a Device 3–2
3.2.2 Disconnecting a Device 3–2
3.2.3 Read and Write Data Functions 3–3

3.2.4 Read and Write Message Functions Using ILAN$DEVICE_SPECIFIC . . . 3–6
3.2.5 SINEC-AP Read and Write Message Functions Using Physical Points . . . 3–8
3.2.5.1 Reading a message . . . 3–9
3.2.5.2 Writing a message . . . 3–9
3.2.5.3 Exchanging a message . . . 3–9
3.2.6 Download, Upload and Directory Functions . . . 3–9
3.2.7 Start and Stop Functions . . . 3–10
3.2.8 Read Status Function . . . 3–12
3.2.8.1 Logical Status Meaning . . . 3–12
3.2.8.2 Physical Status Meaning . . . 3–12
3.3 Automatic Data Collection . . . 3–13
3.3.1 Unsolicited Data . . . 3–13
3.3.1.1 Information Report . . . 3–13
3.3.1.2 Write Indication . . . 3–13
3.3.1.3 Read Indication . . . 3–14
3.3.1.3.1 Automatic Reply to Read Indication . . . 3–14
3.3.1.3.2 Manual Reply to Read Indication . . . 3–14
3.3.1.4 Message Exchange Indication . . . 3–15
3.3.1.4.1 Automatic Reply to Exchange Message Indication . . . 3–15
3.3.1.4.2 Manual Reply to Message Exchange Indication . . . 3–15
3.3.1.5 Local and Remote VMDs . . . 3–16
3.3.2 Pollsets/Scattered Read . . . 3–18

A Returned Values and Associated Error Messages

B Logged Messages

B.1 NI Logged Messages . . . B–1
B.2 PE Logged Messages . . . B–1

Index

Examples

3–1 Connecting to an MMS/SINEC-S7/SINEC-AP/SINEC-H1 Device . . . 3–2
3–2 Disconnecting an MMS/SINEC-S7/SINEC-AP/SINEC-H1 Device . . . 3–3
3–3 Read Data Screen . . . 3–5
3–4 Read Data Screen with Data Array . . . 3–5
3–5 Write Data Screen . . . 3–6
3–6 Write Data Screen with Data Array . . . 3–6
3–7 Read message procedure . . . 3–7
3–8 Write message procedure . . . 3–8
3–9 Directory Screen . . . 3–10
3–10 Upload and Download Screen . . . 3–10
3–11 Read Status Screen . . . 3–12
3–12 Unsolicited Point Definition . . . 3–16

Figures

1–1	DAS Communications	1–2
3–1	Program Invocations State TABLE	3–11
3–2	Scattered Pollset Definition	3–18

Tables

1–1	MMS Client Supported Services	1–3
1–2	Proposed MMS Conformance Building Blocks	1–4
1–3	SINEC-AP and SINEC-H1 Client Supported Services	1–4
1–4	Proposed SINEC-AP and SINEC-H1 Conformance Building Blocks	1–5
1–5	SINEC-S7 Client Supported Services	1–6
1–6	Proposed SINEC-S7 Conformance Building Blocks	1–6
2–1	Disk Space Requirements	2–2
2–2	Files created during installation	2–8
2–3	Path Parameters	2–12
2–4	MMS Application Simple Name attributes	2–15
2–5	Siemens Application Simple Name attributes	2–15
2–6	OSI Transport Template Parameters	2–18
2–7	OSI Transport Parameters	2–18
2–8	BASEstar Classic Parameters	2–19
2–9	SPT Static Block Sizes	2–19
2–10	SPT Dynamic Block Sizes	2–20
3–1	Supported Data Addressing	3–3
3–2	Supported Data Types	3–4
3–3	Supported Address Postfix Values	3–5
3–4	ILAN$DEVICE_SPECIFIC Function Codes	3–6
3–5	Logical Status Meaning	3–12
3–6	Physical Status Meaning	3–13
3–7	Unsolicited ID Values	3–16

Preface

This document describes how to install and use the BASEstar Classic DAS for OMNI software.

Intended Audience

This document is intended for use by system managers who must set up and maintain the following:

- BASEstar Classic for OpenVMS software
- BASEstar Classic DAS for OMNI software

This document is also intended for use by application programmers who develop shop-floor management software layered on BASEstar Classic for OpenVMS.

Readers of this document should have a solid understanding of OpenVMS operations and administration, as well as OpenVMS application software. In addition, knowledge of the OMNI devices and the specific requirements of the installation site is essential.

Document Structure

This document is organized as follows:

- Chapter 1 provides an overview of the DAS for OMNI software.
- Chapter 2 provides information you need to install the DAS for OMNI software.
- Chapter 3 provides information about the supported functions for OMNI devices, and how to access those functions.

Associated Documents

Further information on topics covered in this document can be found in the following documents:

For information on BASEstar Classic see the following documnents:

- BASEstar Classic System Management Documentation, VOLUME 1:
  - *BASEstar Classic Release Notes*
  - *BASEstar Classic Installation Guide*
  - *BASEstar Classic Configuration and Tuning Guide*
- BASEstar Classic General User Documentation, VOLUME 2:
  - *BASEstar Classic Introduction*
  - *BASEstar Classic Menu Interface User's Guide*

  - *BASEstar Classic Command Line Interface User's Guide*
- BASEstar Classic Programming Documentation, VOLUME 3:
  - *BASEstar Classic Introduction to Callable Services*
  - *BASEstar Classic Guide to Writing a Database Server*
  - *BASEstar Classic Guide to Writing Device Access Software*
  - *BASEstar Classic DEC Rdb Database Server Guide*
  - *BASEstar Classic Device Access Software Guide*
- BASEstar Classic Programming Documentation, VOLUME 4:
  - *BASEstar Classic Application Programming Interface Reference Guide*

For information MMS ISO/IEC 9506 see the following documents:

- *Industrial automation systems - Manufacturing Message Specification*
  - *Part 1: Service definition*
  - *Part 1: Service definition*

For information on OMNI software see the following documents:

- *OMNI for OpenVMS Installation Guide*
- *OMNI Definition Facility User's Guide*
- *OMNI Application Interface User's Guide*

For information on OSAP software see the following documents:

- *OSAP/VMS Installation Guide*
- *OSAP/S7 Network Manager's and Programmer's Guide*
- *OSAP/VMS Network Manager's Guide*

For information on DECnet-Plus see the following documentS:

- *DECnet-Plus for OpenVMS Installation and Configuration*
- *DECnet-Plus for OpenVMS Introduction and User's Guide*
- *DECnet-Plus for OpenVMS Network Management*
- *DECnet-Plus Network Control Language Reference*
- *DECnet/OSI for VMS CTF Use*

For information on TCP/IP Services for OpenVMS see the following documentS:

- *TCP/IP Services for OpenVMS Installation and Configuration*
- *TCP/IP Services for OpenVMS User's Guide*
- *TCP/IP Services for OpenVMS Management*

Conventions

This manual uses the following conventions:

Boldface	Highlights user input within textual descriptions.
Return	Press the key labeled Return. Unless otherwise specified, press Return after entering a command or responding to a prompt.
Enter	Type the words or symbols described and press Return.

1
Overview

This chapter provides an overview of the DAS for OMNI software for MMS compliant programmable devices. It also includes a brief description of OMNI device communications, and the supported functions of the DAS for OMNI software.

1.1 Description

The DAS for OMNI software allows you to access devices which implement the Manufacturing Message Specifications (MMS), ISO IS-9506 and/or the Siemens SINEC-S7, SINEC-AP and SINEC-H1 protocols through BASEstar Classic device connection managment. Device connection management is a component of the BASEstar Classic for OpenVMS, which is an engineered family of software components and services that facilitate the development and integration of manufacturing automation solutions.

The BASEstar Classic DAS for OMNI software provides an interface between BASEstar Classic and OMNI software, the Compaq product which implements MMS application protocol. (Under Version 3.0 and following of OMNI software, the OMNI API and MMS protocol are split into separate products.) Furthermore, the DAS for OMNI software provides an interface between BASEstar Classic and the Siemens SINEC-S7/AP/H1 protocols using OSAP/S7, OSAP/AP or OSAP/H1 software, the Compaq products which implements these protocols under the OMNI software architecture.

Using the DAS for OMNI software users or applications can perform a variety of device access functions for devices which implement MMS and/or SINEC-S7/AP /H1 protocols. The DAS for OMNI software receives a request for device access and uses the appropriate sequence of OMNI software calls to forward the request to the device. OMNI software responds to the request and sends the request status back to the DAS.

1.2 Device Communications

The DAS for OMNI software consists of a Protocol Emulator (PE) and a Network Interface (NI).

The PE translates the BASEstar Classic DCM's generic services into calls to the OMNI callable interface, making data access transparent to users.

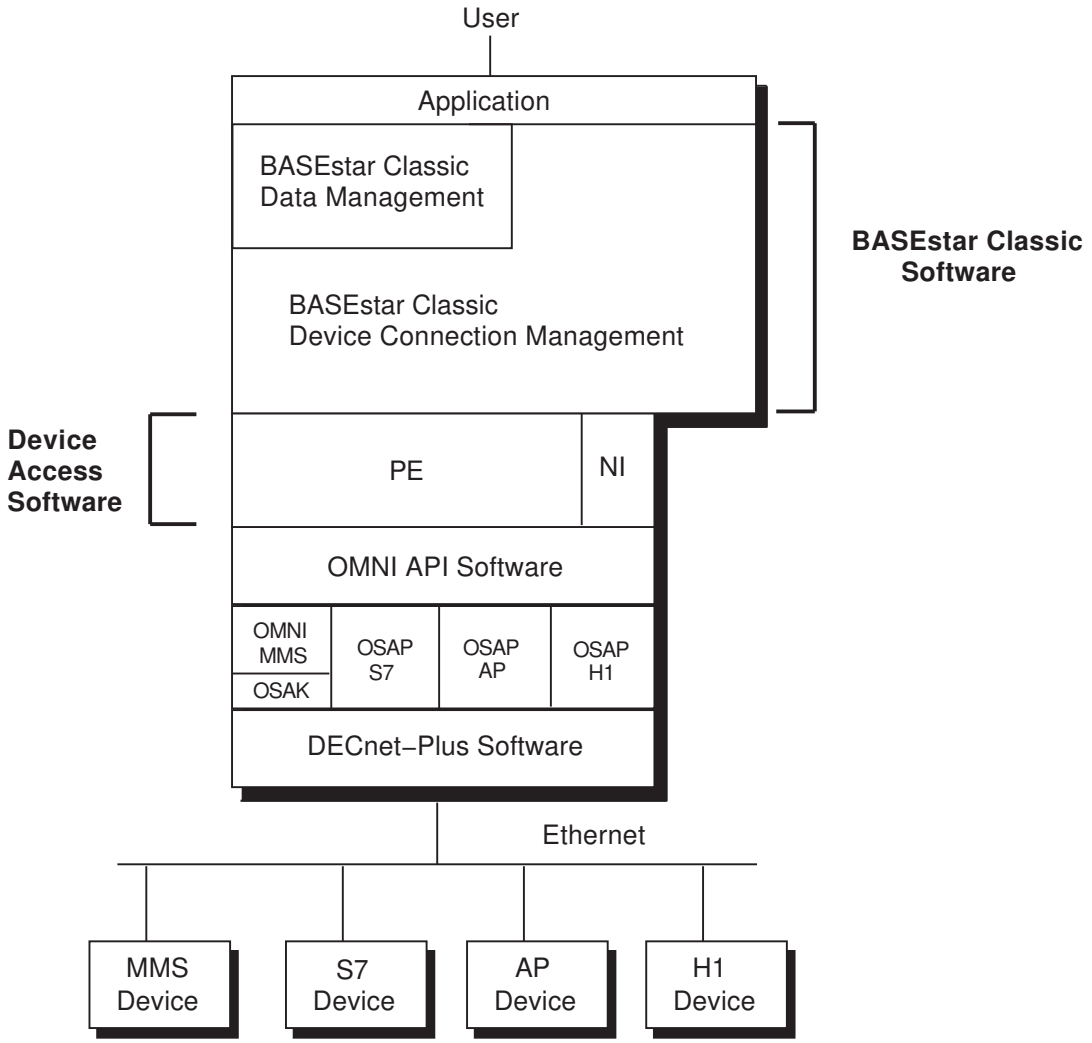
The PE also creates and maintains the OMNI data structures. VMDs, Variable Names and Data Types, are defined using the BASEstar Classic API. Definitions are **not** taken from the OMNI ODF object definitions.

In the DAS for OMNI software, the NI is only responsible for initializing OMNI software.

1.2 Device Communications

The DAS for OMNI software must be installed on an OpenVMS system that is connected to the MMS/SINEC-S7/SINEC-AP/SINEC-H1 device(s) via an 802.3 network. Connections may be made either directly on the same network backbone as the device(s) or indirectly via a bridge, the exact configuration will be dependent on the type of OpenVMS system used, the device network media and the specific networking topology requirements. Figure 1–1 shows how the DAS for OMNI software facilitates communications between device connection management and the devices.

Figure 1–1 DAS Communications



1.3 Supported Functions

You can perform only the BASEstar Classic functions that are supported by a device's PE. These functions can be accessed through the BASEstar Classic software's menu system, commands, and callable services.

1.3.1 Supported MMS Functions

The DAS for OMNI software supports the following BASEstar Classic functions using MMS compliant devices:

- Read and Write Named and Unnamed Variables (Client and Server)
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Read Device Status and Identification
- Receive Unsolicited Status
- Download and Upload Domains (Programs and Data)
- Start and Stop Programs (Program Invocations)
- Get Device Directory (Get Name List)

Table 1–1 lists MMS client supported services. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for a definition of the MMS services.

Table 1–1 MMS Client Supported Services

MMS Service	BASEstar Classic DCM Function
Identify	Read Status
Status	Read Status
GetCapabilityList	Read Status
ReadVariable	Read Data
WriteVariable	Write Data
InformationReport	Unsolicited Data Management
GetNameList	Directory
InitiateDownloadSequence	Download
DownloadSegment	Download
TerminateDownloadSequence	Download
InitiateUploadSequence	Upload
UploadSegment	Upload
TerminateUploadSequence	Upload
CreateProgramInvocation	Start Program
DeleteProgramInvocation	Stop Program
Start	Start Program
Stop	Stop Program
Reset	Stop Program
UnsolicitedStatus	Log Service
Initiate	Enable Device
Conclude	Disable Device
Abort	Disable Device

Table 1–2 lists the MMS conformance building blocks. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for information on conformance building blocks.

Table 1–2 Proposed MMS Conformance Building Blocks

Proposed CBBs	
STR1	Y
STR2	Y
NEST	10
VNAM	Y
VADR	Y
Max Segment Size	512[1]
Version	0/1

[1]This value is read from the IOSIZE attribute on the path definition.

1.3.2 Supported SINEC-AP and SINEC-H1 Functions

The DAS for OMNI software supports the following BASEstar Classic functions using SINEC-AP and SINEC-H1 compliant devices:

- Read and Write Named and Unnamed Variables (Client and Server)
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Read Device Status and Identification
- Receive Unsolicited Status
- Download and Upload Domains (Programs and Data)
- Start and Stop Programs (Program Invocations)
- Get Device Directory (Get Name List)
- Read, Write and Exchange SINEC-AP and SINEC-H1 Messages (Client and Server)

TABLE Table 1–3 lists SINEC-AP and SINEC-H1 client supported services. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for a definition of the MMS services.

Table 1–3 SINEC-AP and SINEC-H1 Client Supported Services

SINEC-AP and SINEC-H1 Service	BASEstar Classic DCM Service
Identify	Read Status
Status	Read Status
GetCapabilityList	Read Status
ReadVariable	Read Data
WriteVariable	Write Data

(continued on next page)

Table 1–3 (Cont.) SINEC-AP and SINEC-H1 Client Supported Services

SINEC-AP and SINEC-H1 Service	BASEstar Classic DCM Service
InformationReport	Unsolicited Device Connect Management
GetNameList	Directory
InitiateDownloadSequence	Download
DownloadSegment	Download
TerminateDownloadSequence	Download
InitiateUploadSequence	Upload
UploadSegment	Upload
TerminateUploadSequence	Upload
CreateProgramInvocation	Start Program
DeleteProgramInvocation	Stop Program
Start	Start Program
Stop	Stop Program
Reset	Stop Program
UnsolicitedStatus	Log Service
Initiate	Enable Device
Conclude	Disable Device
Abort	Disable Device

Table 1–4 lists the SINEC-AP and SINEC-H1 conformance building blocks. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for information on conformance building blocks.

Table 1–4 Proposed SINEC-AP and SINEC-H1 Conformance Building Blocks

Proposed CBBs	
STR1	Y
STR2	Y
NEST	10
VNAM	Y
VADR	Y
Max Segment Size	4096[1]

[1]This value is read from the IOSIZE attribute on the path definition.

1.3.3 Supported SINEC-S7 Functions

The DAS for OMNI software supports the following BASEstar Classic functions using SINEC-S7 compliant devices:

- Read and Write Unnamed Variables (Client and Server)
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Read Device Status and Identification
- Receive Unsolicited Status

TABLE Table 1–5 lists SINEC-S7 client supported services. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for a definition of the MMS services.

Table 1–5 SINEC-S7 Client Supported Services

SINEC-S7 Service	BASEstar Classic DCM Service
Identify	Read Status
Status	Read Status
ReadVariable	Read Data
WriteVariable	Write Data
InformationReport	Unsolicited Device Connect Management
UnsolicitedStatus	Log Service
Initiate	Enable Device
Abort	Disable Device

Table 1–6 lists the SINEC-S7 conformance building blocks. See the *Industrial automation systems - Manufacturing Message Specification Part 1: Service definition* for information on conformance building blocks.

Table 1–6 Proposed SINEC-S7 Conformance Building Blocks

Proposed CBBs	
STR1	Y
STR2	Y
NEST	10
VNAM	N
VADR	Y
Max Segment Size	4096[1]

[1]This value is read from the IOSIZE attribute on the path definition.

1.4 Non-supported Functions

The MMS specification provides a rich variety of device/application functions of which only a sub-set are typically supported by MMS/AP/S7/H1 devices, typically MAP Classes 2 or 3.

The DAS for OMNI software can act as a MMS Server for a subset of MMS functions, ie. read, write and status.

For detailed information about the supported functions, see Chapter 3.

2
Installing and Configuring the DAS

This chapter provides the information you need to install the DAS for OMNI software and to configure your system.

2.1 Installation Requirements

Review the following hardware and software requirements to ensure that your system is prepared for the DAS for OMNI software installation.

Note

Back up the disks on your system before installing this software. This will provide a method to restore your system in the event of an installation problem. The procedure for backing up disks is described in the *OpenVMS System Management Utilities Reference Manual*.

2.1.1 Hardware Requirements

The DAS for OMNI software communicates to MMS devices over a full ISO /OSI seven layer stack using an 802.3 Ethernet Bus network. SINEC-S7/AP /H1 devices communications uses the first four layers of the ISO/OSI stack over an 802.3 Ethernet network. SINEC-S7 also supports communnications using RFC-1006 (OSI over TCP/IP). You will require the appropriate OpenVMS communications network interface and/or bridges to connect the OpenVMS system and MMS/SINEC-S7/SINEC-AP/SINEC-H1 devices together. The same OpenVMS 802.3 Ethernet interface may be used for both DECnet and MMS /SINEC-S7/SINEC-AP/SINEC-H1 communications.

The hardware requirements for the DAS for OMNI software are the same as those for BASEstar Classic. For specific hardware requirements, see the *BASEstar Classic Installation Guide*.

2.1.2 Software Requirements

Before installing the DAS for OMNI software, the following software must already be installed: [1]

- OpenVMS Version 5.5-2 or higher (VAX)
- OpenVMS Version 6.1 or higher (Alpha)
- DECnet/OSI (DECnet-Plus) Version 5.6B or higher (VAX)[1]
- DECnet/OSI (DECnet-Plus) Version 5.7 or higher (Alpha)[1]
- OSI Session Application Kernel (OSAK) Version 3.0

[1] Refer to the *DECnet-Plus for OpenVMS Installation and Configuration* to determine the minimum version of DECnet-Plus required for the version of OpenVMS software installed.

- OMNI for OpenVMS Version 2.0A or higher (VAX)
- OMNI for OpenVMS Version 2.1 or higher (Alpha)
- BASEstar Classic for OpenVMS, Version 3.4 or higher

If BASEstar Classic for OpenVMS is not already installed on your system, see the *BASEstar Classic Installation Guide*.

Note

Before using this product on a system, you must first register a License Product Authorization Key (License PAK) using the License Management Facility (LMF). For more information about the License Management Utility, refer to the *License Management Utility Manual* for OpenVMS.

2.1.3 Additional Software Requirements

In order to support the Siemens SINEC-S7, SINEC-AP or SINEC-H1 functions, the following software must also be installed: [1]

- OSAP/S7 Version 3.1 or higher
- OSAP/AP Version 2.0B or higher[1]
- OSAP/H1 Version 2.0B or higher[1]

If using OMNI software version 3.0 or later, the following software must also be installed to support MMS functions:

- OMNI-MMS Version 3.0 or higher

2.1.4 Disk Space Requirements

Table 2–1 lists the disk space required to install the DAS for OMNI software. The space requirements are approximations; actual sizes may vary depending on your system environment, configuration, and software options installed.

Table 2–1 Disk Space Requirements

Usage	Approximate Space Requirements
Peak (during installation)	1000 blocks (VAX)
	2250 blocks (Alpha)
Net (after installation)	650 blocks (VAX)
	1100 blocks (Alpha)

[1] Refer to the *OSAP/VMS Installation Guide* to determine the minimum version of OMNI or OMNI-API required for the version of OSAP software installed.

2.2 Installation Procedure

When your system meets all the hardware and software requirements, you can install the DAS for OMNI software. The installation takes from 5 to 10 minutes, depending on system load and configuration.

Install the DAS for OMNI software using the following steps:

1. Log in to a privileged system manager's account.
2. Set the default directory to SYS$UPDATE:

```
$  SET DEFAULT SYS$UPDATE
```

3. Invoke VMSINSTAL:

```
$ @SYS$UPDATE:VMSINSTAL DCM_OMNIVVA034 DDCU:
```

   where:

   - **DCM_OMNIVVA034** argument is the kit name
   - **034** portion of the name is the version number
   - **ddcu** argument represents the name of the device on which the installation media is mounted, where:
     - **dd** is the device code
     - **c** is the controller designation
     - **u** is the unit number

VMSINSTAL prompts you for information during the installation. The following is an example of the output from the installation.

```
        OpenVMS VAX Software Product Installation Procedure V7.2

It is 3-FEB-2000 at 08:20.

Enter a question mark (?) at any time for help.

* Are you satisfied with the backup of your system disk [YES]?

The following products will be processed:

  DCM_OMNIVVA V3.4

        Beginning installation of DCM_OMNIVVA V3.4 at 08:21

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.

   Copyright 1990 Compaq Computer Corporation

   Confidential computer software.  Valid license from Compaq or authorized
   sublicensor required for possession, use or copying.  Consistent with FAR
   12.211 and 12.212, Commercial Computer Software, Computer Software
   Documentation, and Technical Data for Commercial Items are licensed to the
   U.S. Government under vendor's standard commercial license.

   BASEstar Classic DAS for OMNI Software installation procedure.

   Checking for a valid license...
```

```
        Product:      DAS-OMNI-CL
        Producer:     DEC
        Version:      3.4
        Release Date: 13-DEC-1996

* Does this product have an authorization key registered and loaded? y

   Now checking OpenVMS version...

   Now checking that BASEstar Classic is installed...

   Now checking that OMNI Software is installed...

   Now checking disk space...

* Do you want to purge files replaced by this installation [YES]?
* Do you want to run the IVP after the installation [YES]?

   The installation procedure has no further questions to ask and will
   complete in 2 to 10 minutes depending on the system and system load.

------------------------------------------------------------------------------
   The configuration template file for OMNI support, DCM_OMNI_CONFIG.TEMPLATE,
   is used to define the OMNI paths, types, and devices.  Edit this file, as
   necessary, to reflect your specific site configuration.

   During installation it will be placed in the directory BCC$SYSDATA.
------------------------------------------------------------------------------

------------------------------------------------------------------------------
   The startup file for ILAN$DEVSRV, ILAN$SYSTEM_STARTUP.COM must be modified
   in order to use the DAS for OMNI Software.  This installation provides a
   substitute template for this file called ILAN$SYSTEM_STARTUP.TEMPLATE with
   modified privileges and quotas.  After installation, edit this file as
   necessary to match your site requirements.  Be sure to restore this edited
   file when upgrading BASEstar Classic software to a new version.

   During installation it will be placed in the directory BCC$SYSDATA.
------------------------------------------------------------------------------

------------------------------------------------------------------------------
   A utility to show the configuration of all OMNI devices,
   DCM_OMNI$SHOW_DEVICE.COM is provided with this DAS.  This utility
   extracts definitions from the BASEstar Classic database as well as from
   ODS and allows you to see the configuration of all your OMNI devices.

   During installation it will be placed in the directory BCC$SYSTEM.
------------------------------------------------------------------------------

   Now linking ILAN_OMNI shared image...

   Now linking IVP test routines...

%VMSINSTAL-I-SYSDIR, This product creates system directory [SYSTEST.DCM_OMNI].
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

   Copyright 1990 Compaq Computer Corporation

   Confidential commercial computer software.  Valid license required.

   Executing the Installation Verification Procedure.

   Now checking OSI version...

   DECnet Plus V7.2 is supported.

   Now checking process privileges...
```

```
   Now checking that files have reached target directories...

   In order to test DAS connections, BASEstar Classic needs to be started.
   BASEstar Classic will be automatically started if you continue this IVP.

* OK to continue IVP? [YES]

   BASEstar Classic will be started, please wait.

   Starting BASEstar Classic, please wait...
   Starting BASEstar Classic, please wait...

   BASEstar Classic is now running.

   Creating objects for MMS runtime test...

   Creating MMS OSI transport template...

Processing NSAP 49::00-22:AA-00-04-00-74-89:20 (LOCAL:.OHF.BASEX2)

   Creating MMS ODS definitions...

   Creating MMS BASEstar Classic objects...

BASEstar V3.4 for VMS
Copyright (c) 1994, Digital Equipment Corporation.  All Rights Reserved.

%BCC-S-OBJCRE, DCM_OMNI$MMS_TYP created.
%BCC-S-OBJCRE, DCM_OMNI$MMS_PTH created.
%BCC-S-OBJCRE, DCM_OMNI$MMS_IVP created.

   Now running the MMS-IS receiver application...

IVP_MMS_RCVR: Starting the IVP receiver.

IVP_MMS_RCVR: listening...

   Now trying to access an MMS device.

   Now writing to the MMS device.

Device      : DCM_OMNI$MMS_IVP
Address     : var_1
Format      : S_LONGWORD

IVP_MMS_RCVR: Accepted the connect.

IVP_MMS_RCVR: Received a write variable request...

   Now reading from the MMS device.

IVP_MMS_RCVR: Received a read variable request...

Device      : DCM_OMNI$MMS_IVP
Address     : var_1
Format      : S_LONGWORD

Data:
    0       :            10

IVP_MMS_RCVR: received an ABORT.

IVP_MMS_RCVR: All done...

   Creating objects for SINEC-AP runtime test...

   Creating SINEC-AP OSI transport template...

Routing inactive area defined = {49::FF-00}
Local SINEC-AP/H1 address = AA0004007489
```

```
   Creating SINEC-AP ODS definitions...

   Creating SINEC-AP BASEstar Classic objects...

BASEstar V3.4 for VMS
Copyright (c) 1994, Digital Equipment Corporation.  All Rights Reserved.

%BCC-S-OBJCRE, DCM_OMNI$AP_TYP created.
%BCC-S-OBJCRE, DCM_OMNI$AP_PTH created.
%BCC-S-OBJCRE, DCM_OMNI$AP_IVP created.

   Now running the SINEC-AP receiver application...

IVP_AP_RCVR: Starting the IVP receiver.

IVP_AP_RCVR: listening...

   Now trying to access a SINEC-AP device.
IVP_AP_RCVR: Accepted the connect.

   Now writing to the SINEC-AP device.

Device      : DCM_OMNI$AP_IVP
Address     : var_1
Format      : S_LONGWORD


IVP_AP_RCVR: Received a write variable request...

   Now reading from the SINEC-AP device.

IVP_AP_RCVR: Received a read variable request...

Device      : DCM_OMNI$AP_IVP
Address     : var_1
Format      : S_LONGWORD

Data:
    0       :            10

IVP_AP_RCVR: received an ABORT.

IVP_AP_RCVR: All done...

   Creating objects for SINEC-H1 runtime test...

   Creating SINEC-H1 OSI transport template...

Routing inactive area defined = {49::FF-00}
Local SINEC-AP/H1 address = AA0004007489

   Creating SINEC-H1 ODS definitions...

   Creating SINEC-H1 BASEstar Classic objects...

BASEstar V3.4 for VMS
Copyright (c) 1994, Digital Equipment Corporation.  All Rights Reserved.

%BCC-S-OBJCRE, DCM_OMNI$H1_TYP created.
%BCC-S-OBJCRE, DCM_OMNI$H1_PTH created.
%BCC-S-OBJCRE, DCM_OMNI$H1_IVP created.

   Now running the SINEC-H1 receiver application...

IVP_H1_RCVR: Starting the IVP receiver.

IVP_H1_RCVR: listening...

   Now trying to access a SINEC-H1 device.
IVP_H1_RCVR: Accepted the connect.

   Now writing to the SINEC-H1 device.
```

```
Device      : DCM_OMNI$H1_IVP
Address     : U\DB:190:7
Format      : S_LONGWORD


IVP_H1_RCVR: Received a write variable request...

   Now reading from the SINEC-H1 device.

IVP_H1_RCVR: Received a read variable request...

Device      : DCM_OMNI$H1_IVP
Address     : U\DB:190:7
Format      : S_LONGWORD

Data:
    0       :            10

IVP_H1_RCVR: received an ABORT.

IVP_H1_RCVR: All done...

   Creating objects for Siemens-S7 runtime test...


   Creating Siemens-S7 OSI transport template...

Routing inactive area defined = {49::FF-00}
Local SINEC-AP/H1 address = AA0004007489

   Creating Siemens-S7 ODS definitions...


   Creating Siemens-S7 BASEstar Classic objects...

BASEstar V3.4 for VMS
Copyright (c) 1994, Digital Equipment Corporation.  All Rights Reserved.

%BCC-S-OBJCRE, DCM_OMNI$S7_TYP created.
%BCC-S-OBJCRE, DCM_OMNI$S7_PTH created.
%BCC-S-OBJCRE, DCM_OMNI$S7_IVP created.

   Now running the SINEC-S7 receiver application...


IVP_S7_RCVR: Starting the IVP receiver.

IVP_S7_RCVR: listening...

   Now trying to access a SINEC-S7 device.
IVP_S7_RCVR: Accepted the connect.

   Now writing to the SINEC-S7 device.


Device      : DCM_OMNI$S7_IVP
Address     : U\DB:190:7
Format      : S_LONGWORD


IVP_S7_RCVR: Received a write variable request...

   Now reading from the SINEC-S7 device.

IVP_S7_RCVR: Received a read variable request...

Device      : DCM_OMNI$S7_IVP
Address     : U\DB:190:7
Format      : S_LONGWORD

Data:
    0       :            10

IVP_S7_RCVR: received an ABORT.
```

```
IVP_S7_RCVR: All done...
   The BASEstar Classic DAS for OMNI Software IVP has successfully completed.

        Installation of DCM_OMNIVVA V3.4 completed at 08:25

        VMSINSTAL procedure done at 08:25
```

2.2.1 Files Created During Installation

Table 2–2 lists the files created by the DAS for OMNI software installation procedure, and the directories in which those files are placed.

Table 2–2 Files created during installation

Directory	Filename
BCC$SYSDATA:	ILAN$SYSTEM_STARTUP.TEMPLATE
	DCM_OMNI_CONFIG.TEMPLATE
BCC$SYSTEM:	DCM_OMNI$SHOW_DEVICE.COM
SYS$LIBRARY:	ILAN_OMNI.EXE
	DCM_OMNI$DEF.H
SYS$HELP:	DCM_OMNIVVA034.RELEASE_NOTES (VAX)
	DCM_OMNIVAA034.RELEASE_NOTES (Alpha)
SYS$TEST:	DCM_OMNI$IVP.COM
SYS$COMMON:[SYSTEST.DCM_OMNI]	DCM_OMNI$IVP_MMS_RCVR.EXE
	DCM_OMNI$IVP_S7_RCVR.EXE
	DCM_OMNI$IVP_AP_RCVR.EXE
	DCM_OMNI$IVP_H1_RCVR.EXE
	DCM_OMNI$STOP_DCM.EXE

2.2.2 Installation Messages

You may see VMSINSTAL messages during the installation procedure.

BADBCC, BASEstar Classic software must be installed before the DAS for OMNI Software.

Explanation: Error. Incorrect version of or missing BASEstar Classic software.

User Action: Install BASEstar Classic for OpenVMS, Release 3.4 or higher software.

BADDCM, BASEstar Classic Device Connect must be installed before the DAS for OMNI Software.

Explanation: Error. Incorrect version of or missing BASEstar Classic DCM software.

User Action: Install BASEstar Classic DCM for OpenVMS, Release 3.4 or higher software.

BADOMNI (VAX), The DAS for OMNI Software must be installed under OMNI 2.0A or greater.

Explanation: Error. An incorrect version of OMNI for OpenVMS was installed on this system.

User Action: Install OMNI for OpenVMS V2.0A or higher and then reinstall the DAS for OMNI software.

BADOMNI (Alpha), The DAS for OMNI Software must be installed under OMNI 2.1 or greater.

Explanation: Error. An incorrect version of OMNI for OpenVMS was installed on this system.

User Action: Install OMNI for OpenVMS V2.1 or higher and then reinstall the DAS for OMNI software.

BADVMS (VAX), The DAS for OMNI Software must be installed under OpenVMS V5.5-2 or greater.

Explanation: Error. Incorrect version of OpenVMS.

User Action: Install OpenVMS V5.5-2 or higher.

BADVMS (Alpha), The DAS for OMNI Software must be installed under OpenVMS V6.1 or greater.

Explanation: Error. Incorrect version of OpenVMS.

User Action: Install OpenVMS V5.5-2 or higher.

BCCNORUN, BASEstar Classic will be started, please wait.

Explanation: Informational. The installation procedure is starting BASEstar Classic.

User Action: None.

BCCSTART, Starting BASEstar Classic, please wait ...

Explanation: Informational. The installation procedure has started BASEstar Classic and is waiting for the status to go to "running".

User Action: None.

BSTARRUN, BASEstar Classic is now running.

Explanation: Informational. The installation procedure has completed starting BASEstar Classic.

User Action: None.

NETBLOCKS (VAX), The BASEstar Classic DAS for OMNI V3.4 requires 650 blocks after installation.

Explanation: Error. Not enough disk space to complete installation.

User Action: Delete any unnecessary files, then reinstall.

NETBLOCKS (Alpha), The BASEstar Classic DAS for OMNI V3.4 requires 1100 blocks after installation.

Explanation: Error. Not enough disk space to complete installation.

User Action: Delete any unnecessary files, then reinstall.

NOOMNI, Your system does not have OMNI for OpenVMS installed

Explanation: Error. OMNI for OpenVMS must be installed before the DAS for OMNI Software.

User Action: Install OMNI for OpenVMS and then install the DAS for OMNI software.

NORDB, You have Multi-version RDB but the current RDB version is not set, please issue @SYS$LIBRARY:RDBVMS_SETVER.COM and repeat the DAS for OMNI Software installation.

Explanation: Error. Don't know which SQL library to link against.

User Action: Execute @SYS$LIBRARY:RDBVMS_SETVER.COM and repeat the DAS for OMNI Software installation.

NOLICENSE, No license found for this product - IVP will not be run.,

Explanation: Informational. A valid license was not found. The installation will continue, but the IVP will not be run.

User Action: Register and load a valid license for this product before attempting to use the DAS.

NOLOAD, License for this product not loaded - IVP will not be run.,

Explanation: Informational. The license for this product has not been loaded by the License Management Utility. The installation willl proceed, but the IVP will not be run.

User Action: Load the license using the License Management Utility before attempting to use the DAS.

2.3 Post Installation Tasks

This section describes the tasks to perform after installing the DAS for OMNI software, including editing the configuration file, setting the BASEstar Classic support block parameter, and setting up shop floor equipment.

2.3.1 Configuring layered products

See the related OMNI/OSAP and DECnet-Plus documentation for configuring the layered products used by the DAS for OMNI software. See Section 2.3.2.4 for information on configuring OSI transport templates.

2.3.2 Editing the Configuration File

A configuration file is supplied with the DAS for OMNI software. The configuration file contains definitions for types, paths, and devices.

- Type record—represents a Protocol Emulator (PE)
- Path record—represents a Network Interface (NI)
- Device record—represent a device.

The following sections give examples of the type, path, and device records. See the *BASEstar Classic Command Line Interface User's Guide* for more information about creating type, path, and device definitions.

2.3.2.1 Editing Type Records

The following example shows the type records created by the configuration file.

```
create type MMS_IS_TYPE/manufacturer=DEC/model="MMS-IS\MMS-IS device"-
   /protocol=OMNI -
   /description="MMS-IS ISO 9506 compliant device"/log

create type MMS_DIS_TYPE/manufacturer=DEC/model="MMS-DIS\MMS-DIS device"-
   /protocol=OMNI -
   /description="MMS-DIS ISO 9506 compliant device"/log

create type AP_TYPE/manufacturer="Siemens-AG"/model="AP\Siemens AP device"-
   /protocol=OMNI -
   /description="Siemens SINEC-AP device"/log

create type H1_S115U_941_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_115U-941"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S115U_942_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_115U-942"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S115U_943_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_115U-943"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S115U_944_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_115U-944"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S135U_921_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_135U-921"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S135U_922_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_135U-922"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S135U_928_TYP/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_135U-928"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S150U_TYPE/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_150U"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type H1_S155U_TYPE/manufacturer="Siemens-AG"-
   /model="H1\SIMATIC_155U"-
   /protocol=OMNI -
   /description="Siemens SINEC-H1 device"/log

create type S7_TYPE/manufacturer="Siemens-AG"-
   /model="S7\SIMATIC_S7"-
   /protocol=OMNI -
   /description="Siemens S7 device"/log
```

The model is used to define the protocol profile used by the DAS in defining the device to OMNI. The DAS for OMNI software currently supports MMS-IS, MMS-DIS, S7, AP and H1 protocols.

The text before the \ is not modifiable. The text after the \ is used in defining the model to OMNI software. For SINEC-H1 devices, the model field is not modifiable.

2.3.2.2 Editing Path Records

The following example shows the path records included in the configuration file. The commands are commented out in the configuration file so edit the commands and then remove the comment character before executing the configuration file.

```
create path MMS_IS_LCL_PATH/vaxport="MMS_IS_LOCAL_PLC"/netname=OMNI-
/multidrop/io_size=512/log

create path MMS_DIS_LCL_PATH/vaxport="MMS_DIS_LOCAL_PLC"/netname=OMNI-
/multidrop/io_size=512/log

create path AP_LOCAL_PATH/vaxport="S5_AP_LOCAL_PLC"/netname=OMNI-
/multidrop/io_size=4096/log

create path H1_LOCAL_PATH/vaxport="S5_H1_LOCAL_PLC"/netname=OMNI-
/multidrop/io_size=4096/log

create path S7_LOCAL_PATH/vaxport="S7_LOCAL_PLC"/netname=OMNI-
/multidrop/io_size=4096/log
```

The VAXPORT field contains the local Application Simple Name. This application simple name must be defined in ODS (OMNI Directory Services) to define the network address information for the local VMD. Only one path is required to be defined for each protocol profile. However, multiple paths may be defined if it is desired to define a unique local VMD for each remote VMD. See the discussion in Section 3.3 for more information on defining multiple local paths.

Table 2–3 lists the path parameters and indicates the value(s) that are allowed for each.

Table 2–3 Path Parameters

Parameter	Value(s)
VAXport[1]	ASN
Netname	OMNI
Multidrop[2]	MULTIDROP, POINT_TO_POINT
Timeout[3]	
Retries[4]	
IO Size[5]	

[1]This parameter must be set to the Application Simple Name defined in ODS for the local VMD.

[2]This parameter is set to "POINT_TO_POINT" if a unique path is defined for each device.

[3]The parameter is ignored by this DAS. The timeout for an I/O operation is controlled by the OSI transport template parameters.

[4]The parameter is ignored by this DAS. The DAS does not do retries.

[5]The parameter is used to set the maximum segment value when defining the device to OMNI software. See the documentation for your device to set this parameter. If not specified, this parameter defaults to 512.

The line parameters in the path definition are not used by this DAS and are ignored.

2.3.2.3 Editing Device Records

The following example shows the device records created by the configuration file.

NOTE: The device MUST be defined with the UNSOLICITED attribute.

```
create dev MMS_IS_PLC_NO_01/path=MMS_IS_LCL_PATH/type=MMS_IS_TYPE-
/netaddr=MMS_IS_PLC_NO_01/unsol/timeout=600/log

create dev MMS_DIS_PLC_NO_01/path=MMS_DIS_LCL_PATH/type=MMS_DIS_TYPE-
/netaddr=MMS_DIS_PLC_NO_01/unsol/timeout=600/log

create dev S5_AP_PLC_NO_01/path=AP_LOCAL_PATH/type=AP_TYPE-
/netaddr=S5_AP_PLC_NO_01/unsol/timeout=600/log

create dev S5_H1_PLC_NO_01/path=H1_LOCAL_PATH/type=H1_S150U_TYPE-
/netaddr=S5_H1_PLC_NO_01/unsol/timeout=600/log

create dev S7_PLC_NO_01/path=S7_LOCAL_PATH/type=S7_TYPE-
/netaddr=S7_PLC_NO_01/unsol/timeout=600/log
```

The device name is the name of a remote VMD. Remote VMD names and related definitions (variables, domains, etc.) need not be defined in the OMNI database. They are created at run-time by the DAS for OMNI software and are taken from the BASEstar Classic definitions ONLY.

NOTE

The device definition MUST be defined as UNSOLICITED. Failure to do so will prevent the DAS for OMNI software from being able to react to Abort or Conclude indications sent either by the target device or the local OSI network. ie. the DAS for OMNI software will be unaware that its connection with the remote device has been terminated. Recovery will require that the user disable and re-enable the device after which a connection will be re-established. Furthermore, failure to specify support for unsolicited data means that the DAS for OMNI software will buffer all unsolicited data from the device. This will eventually lead to utilization of all the available OpenVMS virtual memory. Subsequently the DAS for OMNI software and ILAN will be unable to continue and will automatically shut themselves down.

The timeout on the device definition controls the time that BASEstar Classic device connection management allows for a device operation to complete. The value for the device timeout should be larger than the expected time of the longest device operation and also longer than the network timeout. The network timeout is controlled by the keepalive timer and retransmit threshold on the OSI transport template. For Siemens devices the RESPTIMER and COMPLTIMER in the ODS configuration can also be used to time out requests.

The local and remote Application Simple Names must be configured with ODSCL. The following examples shows the ODSCL commands contained in the configuration file.

```
$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=MMS_IS_LOCAL_PLC"
REGISTER DIRECTORY NAME "/CN=MMS_IS_LOCAL_PLC"-
ATTRIBUTE "/OC=TAE/AEQ=11/APC={1 0 9506 2 3}-
/P_ADDR=0x01.0x01.0x01.INTERNET%49003FAA00040013FD21"
EXIT
```

```
$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=MMS_IS_PLC_NO_01"
REGISTER DIRECTORY NAME "/CN=MMS_IS_PLC_NO_01"-
ATTRIBUTE "/OC=TAE/AEQ=11/APC={1 0 9506 2 3}-
/P_ADDR=0x02.0x02.0x02.INTERNET%49003FAA00040013FD21"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=MMS_DIS_LOCAL_PLC"
REGISTER DIRECTORY NAME "/CN=MMS_DIS_LOCAL_PLC"-
ATTRIBUTE "/OC=TAE/AEQ=11/APC={1 0 9506 1 1}-
/P_ADDR=0x01.0x01.0x01.INTERNET%49003FAA00040013FD21"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=MMS_DIS_PLC_NO_01"
REGISTER DIRECTORY NAME "/CN=MMS_DIS_PLC_NO_01"-
ATTRIBUTE "/OC=TAE/AEQ=11/APC={1 0 9506 1 1}-
/P_ADDR=0x02.0x02.0x02.INTERNET%49003FAA00040013FD21"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=S5_AP_LOCAL_PLC"
REGISTER DIRECTORY NAME "/CN=S5_AP_LOCAL_PLC"-
ATTRIBUTE "/OC=OSAP/TSAP=OSAPHOST-
/NETADDRESS=IEEE%AA00040013FD"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=S5_AP_PLC_NO_01"
REGISTER DIRECTORY NAME "/CN=S5_AP_PLC_NO_01"-
ATTRIBUTE "/OC=OSAP/TSAP=OSAPTSAP-
/NETADDRESS=IEEE%AA00040013FD"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=S5_H1_LOCAL_PLC"
REGISTER DIRECTORY NAME "/CN=S5_H1_LOCAL_PLC"-
ATTRIBUTE "/OC=OSH1/TSAP={R:SH1HOSTR.W:SH1HOSTW.M:SH1HOSTM}-
/NETADDRESS=IEEE%AA00040013FD"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=S5_H1_PLC_NO_01"
REGISTER DIRECTORY NAME "/CN=S5_H1_PLC_NO_01"-
ATTRIBUTE "/OC=OSH1/TSAP={R:SH1TSAPR.W:SH1TSAPW.M:SH1TSAPM}-
/NETADDRESS=IEEE%AA00040013FD"
EXIT

$ RUN ODS:[EXE]ODSCL
DEREGISTER DIRECTORY NAME "/CN=S7_PLC_NO_01"
REGISTER DIRECTORY NAME "/CN=S7_PLC_NO_01"-
ATTRIBUTE "/OC=OSAPS7/TSAP=%X0103-
/NETADDRESS=IEEE%080006013431"
EXIT
```

Table 2–4 lists the attributes for ODS Application Simple Names for MMS-IS and MMS-DIS Application profiles.

Table 2–4 MMS Application Simple Name attributes

Application Profile	Attribute	Description	Value
MMS-IS	OC	Object Class	TAE
	CN[1]	Common Name	
	APT[23]	Application Process Title	
	AEQ[23]	Application Entity Qualifier	
	APC	Application Context	{1 0 9506 2 3}
	P_ADDR[43]	Presentation Address	
MMS-DIS	OC	Object Class	TAE
	CN[1]	Common Name	
	APT[23]	Application Process Title	
	AEQ[23]	Application Entity Qualifier	
	APC	Application Context	{1 0 9506 1 1}
	P_ADDR[43]	Presentation Address	

[1]The common name entry must equal the name specified in the corresponding VAXport definition.

[2]The APT, AEQ and P_ADDR must be unique for a given node. APT and AEQ are optional attributes. If APT and AEQ are not specified, then OMNI uses the P_ADDR only in matching incoming initiate requests.

[3]See the vendor documentation for the remote VMD to see how this item is defined in the remote VMD.

[4]The presentation address consists of the PSAP, SSAP, TSAP & NSAP. The NSAP consists of the OSI transport template and the OSI network address of the remote VMD.

Table 2–5 lists the attributes for ODS Application Simple Names of SINEC-S7, SINEC-AP and SINEC-H1 Application Profiles.

Table 2–5 Siemens Application Simple Name attributes

Application Profile	Attribute	Description	Value
Sinec S7	OC	Object Class	OSAPS7
	CN[1]	Common Name	
	TSAP[2]	Transport Service Access Point	
	NETADDRESS[3]	Network Address	
	RESPTIMER[4]	Response Timeout	
	COMPLTIMER[5]	Completion Timeout	

[1]The common name entry must equal the name specified in the corresponding VAXport definition.

[2]The TSAP is predefined and is dependent on the positioning of either the communication card or CPU in the rack and slot. See the Siemens S7 documentation for your CPU for details.

[3]The network address consists of the null internet OSI template and the ethernet address of the remote VMD. If using RFC-1006, the network address consists of the RFC-1006 OSI template and the TCP/IP address of the remote VMD.

[4]Timeout for a response from the device. See OSAP documentation for details.

[5]Timeout for completion of a request. The RESPTIMER is usually a submultiple of this value. See OSAP documentation for details.

(continued on next page)

Table 2–5 (Cont.) Siemens Application Simple Name attributes

Application Profile	Attribute	Description	Value
Sinec AP	OC	Object Class	OSAP
	CN[1]	Common Name	
	TSAP[6]	Transport Service Access Point	
	NETADDRESS[7]	Network Address	
	RESPTIMER[4]	Response timeout	
	COMPLTIMER[5]	Completion timeout	
	MUX[8]		
	NEGOTIATION[8]		
Sinec H1	OC	Object Class	OSH1
	CN[1]	Common Name	
	TSAP[9]	Transport Service Access Point	
	NETADDRESS[7]	Network Address	
	RESPTIMER[4]	Response Timeout	
	COMPLTIMER[5]	Completion Timeout	

[1]The common name entry must equal the name specified in the corresponding VAXport definition.

[4]Timeout for a response from the device. See OSAP documentation for details.

[5]Timeout for completion of a request. The RESPTIMER is usually a submultiple of this value. See OSAP documentation for details.

[6]See the vendor documentation for the remote VMD to see how this item is defined in the remote VMD.

[7]The network address consists of the null internet OSI template and the ethernet address of the remote VMD.

[8]Optional parameter. See OSAP documentation for details.

[9]The TSAPs in the remote VMD should equal xxxxxxxR, xxxxxxxW, xxxxxxxM for read, write and message access points.

Make sure that the ODSCL attributes match your shop-floor configuration.

2.3.2.4 NCL Configuration

In order to use the DAS for OMNI software OSI transport templates need to be provided for MMS, S7, AP and H1 connections. To configure OSI transport templates use the command procedure @SYS$STARTUP:NET$CONFIGURE.COM ADVANCED. Select Configure Transports from the menu selection and follow the instructions for configuring the templates. For Sinec S7, Sinec AP and Sinec H1, a NULL_INTERNET template must be configured. For MMS, an INTERNET template is configured. For details on configuring templates, see *DECnet-Plus for OpenVMS Installation and Configuration*.

For Sinec-S7 over TCP/IP an RFC-1006 template is configured. In addition running OSI Applications over TCP/IP must be enabled using @SYS$STARTUP:NET$CONFIGURE.COM ADVANCED and the PWIP driver must be configured using @SYS$STARTUP:UCX$CONFIG.COM. For details on configuring UCX (TCP/IP), see *TCP/IP Services for OpenVMS Installation and Configuration*.

The following is an example of an NCL configuration script showing creation of transport templates for MMS, S7, AP and H1 protocols.

```
CREATE NODE 0 OSI TRANSPORT TEMPLATE OMNINET
SET    NODE 0 OSI TRANSPORT TEMPLATE OMNINET NETWORK SERVICE CLNS, -
CLASSES {4}, -
CONS TEMPLATE , -
EXPEDITED DATA TRUE, -
CHECKSUMS TRUE, -
INBOUND TRUE, -
LOOPBACK FALSE
CREATE NODE 0 OSI TRANSPORT TEMPLATE S7NET
SET    NODE 0 OSI TRANSPORT TEMPLATE S7NET NETWORK SERVICE CLNS, -
CLASSES {4}, -
CONS TEMPLATE , -
EXPEDITED DATA TRUE, -
CHECKSUMS TRUE, -
INBOUND TRUE, -
LOOPBACK FALSE
SET    NODE 0 OSI TRANSPORT TEMPLATE S7NET CLNS INACTIVE AREA ADDRESS {49::FF-00}
SET NODE 0 ROUTING CIRCUIT CSMACD-0 INACTIVE AREA ADDRESS = {49::FF-00}
CREATE NODE 0 OSI TRANSPORT TEMPLATE APNET
SET    NODE 0 OSI TRANSPORT TEMPLATE APNET NETWORK SERVICE CLNS, -
CLASSES {4}, -
CONS TEMPLATE , -
EXPEDITED DATA TRUE, -
CHECKSUMS TRUE, -
INBOUND TRUE, -
LOOPBACK FALSE
SET    NODE 0 OSI TRANSPORT TEMPLATE APNET CLNS INACTIVE AREA ADDRESS {49::FF-00}
SET NODE 0 ROUTING CIRCUIT CSMACD-0 INACTIVE AREA ADDRESS = {49::FF-00}
CREATE NODE 0 OSI TRANSPORT TEMPLATE H1NET
SET    NODE 0 OSI TRANSPORT TEMPLATE H1NET NETWORK SERVICE CLNS, -
CLASSES {4}, -
CONS TEMPLATE , -
EXPEDITED DATA TRUE, -
CHECKSUMS FALSE, -
INBOUND TRUE, -
LOOPBACK FALSE
SET    NODE 0 OSI TRANSPORT TEMPLATE H1NET CLNS INACTIVE AREA ADDRESS {49::FF-00}
SET NODE 0 ROUTING CIRCUIT CSMACD-0 INACTIVE AREA ADDRESS = {49::FF-00}
ENABLE NODE 0 OSI TRANSPORT
CREATE NODE 0 SESSION CONTROL TRANSPORT SERVICE OSI PROTOCOL = %X05
```

S7 devices also support OSI communications over TCP/IP using RFC-1006. The following is an example of an NCL configuration script showing creation of an RFC-1006 transport template for the S7 protocol.

```
CREATE NODE 0 OSI TRANSPORT TEMPLATE S7NET
SET    NODE 0 OSI TRANSPORT TEMPLATE S7NET NETWORK SERVICE RFC1006, -
CLASSES {0,2}, -
EXPEDITED DATA TRUE, -
CHECKSUMS FALSE, -
INBOUND TRUE, -
RFC1006 PORT NUMBER 102, -
LOOPBACK FALSE
ADD NODE 0 OSI TRANSPORT RFC1006 LISTENER PORT {102}
```

Table 2–6 gives a list of recommended OSI transport template parameters. Table 2–7 gives a list of OSI transport parameters that may need to be modified for your intallation. These parameters are explained in detail in the DECnet-Plus documentation. Some are inter-dependent so that modifying one parameter may require changing other ones also.

Table 2–6 OSI Transport Template Parameters

Parameter	Recommended Value[1]
Keepalive Time	3[2]
Retransmit Threshold	2[2]
CR Timeout	5
ER Timeout	5

[1]The values of these parameters depends on your specific network configuration and expected response times.

[2]The product of these values must be lower than the /TIMEOUT value on the device definition.

Table 2–7 OSI Transport Parameters

Parameter
Maximum Transport Connections
Maximum Remote NSAPs

Other timers in the transport template may need to be modified depending on your specific hardware and network configuration.

To modify transport template values place an NCL script in SYS$COMMON:[SYSMGR] and execute it from your system startup command file. Use the following syntax to execute an NCL script.

```
$RUN SYS$SYSTEM:NCL
@SYS$MANAGER:CUSTOM_NCL_SCRIPT.NCL
```

2.3.3 Tuning ILAN$DEVSRV

The installation procedure provides a replacement for the ILAN startup file, ILAN$SYSTEM_STARTUP.TEMPLATE. The new file has been provided because the ILAN$DEVSRV image using the DAS for OMNI software needs additional quotas and privileges. The values used in the provided startup file are valid for a typical configuration and may be more or less than is actually required for your installation.

Note

If quotas are not set properly, it is likely that the ILAN$DEVSRV process will ACCVIO or otherwise fail with very little diagnostic information provided. If you find that the ILAN$DEVSRV process is ACCVIOing soon after enabling devices or soon after startup, monitor the process using `SHOW PROCESS/QUOTA/ID=XXXXXXXX` to determine which quota the process is running low on.

SYSGEN parameters may also need adjusting. Parameters which may need modifying include the following:

- CHANNELCNT
- VIRTUALPAGECNT

BASEstar Classic parameters may also need adjusting, depending on the number of devices configured. Parameters which may need modifying include the following:

- ILAN$MAX_DEVICES
- ILAN$MAX_OUT_REQUESTS
- ILAN$MAX_SPT_REQUESTS
- ILAN$MAX_DSEC_SIZE

See *BASEstar Classic Configuration and Tuning Guide* for details on changing BASEstar Classic parameters.

2.3.4 Defining OMNI Constants

A BASEstar Classic parameter is used to configure the maximum number of OMNI devices that can be created. This parameter is created as part of the DAS installation. To change the value of this parameter, change the value using the BASEstar Classic CLI. Then stop and restart BASEstar Classic to to enable the DAS to use the new parameter value. Table 2–8 lists the default and maximum value for this parameter.

Table 2–8 BASEstar Classic Parameters

Parameter	Default Value	Max Value
ILAN$OMNI_MAX_DEVICES	64	500

2.3.5 Setting the SPT Block Parameter

The ILAN$MAX_SPT_REQUESTS parameter specifies the total number of blocks that can be allocated in the SPT (support) global section. DASes use blocks in the global section for storing data structures and for doing device I/O. The SPT global section is sized by calculating the number of SMALL, MEDIUM, LARGE and EXTRA LARGE blocks that the section should contain. Some blocks remain for the life of a device and some are allocated and deallocated for each I/O operation. Table 2–9 shows the static blocks of each size that are used by the DAS.

Table 2–9 SPT Static Block Sizes

Block Size	Quantity[1]	Block Type
MEDIUM	2	Device
SMALL	1	Device[2]
	1	Unsolicited point[2]
	1	DAS

[1]Quantity is quantity per device, per point, etc.

[2]Only created if the device is marked "unsolicited".

Table 2–10 shows the dynamic blocks of each size that are used by the DAS. These blocks are created and deleted as the device does I/O. The DAS also allocates and deallocates virtual memory to store control information when doing device I/O.

Table 2–10 SPT Dynamic Block Sizes

Block Size	Quantity[1]	I/O Type
SMALL	1	Any

[1]Quantity is quantity per I/O.

The size of the SPT global section can be tuned by changing the percentage of each kind of block that is created. Refer to the *BASEstar Classic Configuration and Tuning Guide* for instructions on changing the percentage of each size of block that is created in the global section.

2.3.6 Setting Up Shop Floor Equipment

To set up your shop floor equipment, see the documentation for your specific MMS, SINEC-S7, SINEC-AP or SINEC-H1 compliant device.

When configuring MMS devices be sure to note the PSAP, SSAP, TSAP and NSAP.

When configuring SINEC-S7, SINEC-AP or SINEC-H1 devices be sure to note the TSAP.

2.4 Tracing OSI Messages

The tracing facility enables you to see if, and what actually is send out and received from OpenVMS on the OSI stack.

You could invoke the Common Trace Facility for a full and live trace by typing:

```
$ TRACE
CTF> START/LIVE/FULL "OSITP CR MESSAGES *"
```

The above trace will trace all connect request protocol data units.

See *DECnet/OSI for VMS CTF Use* for a further description of the Common Trace Facility, tracepoints supported and the output generated.

2.5 Failures During Product Use

If an error occurs while this product is in use and you believe the error is caused by a problem with the product, take one of the following actions:

- If you have a Software Product Services Support Agreement, contact your Customer Support Center (CSC) by telephone or by using the electronic means provided with your support agreement (such as DSNlink). The CSC provides telephone support for high-level advisory and remedial assistance. When you initially contact the CSC, indicate the following:
  - The name and version number of the operating system you are using
  - The version number of the product you are using
  - The version number of BASEstar Classic you are using
  - The version number of OMNI software you are using

  - The version number of OSAP software you are using (if appropriate)
  - The version number of DECnet-Plus (DECnet/OSI) you are using
  - The hardware system you are using (such as a model number)
  - The manufacturer and model numbers of the devices you are communicating with
  - A brief description of the problem (one sentence if possible)
  - How critical the problem is

- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you do not have any type of software services support agreement and you purchased this product within the past year, you can submit an SPR if you think the problem is caused by a software error.

When you submit an SPR, take the following steps:

1. Describe as accurately as possible the circumstances and state of the system when the problem occurred. Include the description and version number of the product being used. Demonstrate the problem with specific examples.
2. Reduce the problem to as small a size as possible.
3. Remember to include listings of any command files, INCLUDE files, or relevant data files, and so forth.
4. Report only one problem per SPR. This will facilitate a faster response.
5. Mail the SPR package to Compaq.

3
Using the DAS

This chapter provides information about the supported functions for OMNI devices, and how to access these functions.

3.1 Accessing OMNI Functions

The DAS for OMNI software functions are accessed through the BASEstar Classic, using the following interfaces:

- Commands
- Menu system
- Callable services

To use the BASEstar Classic services, enter the following command at the DCL prompt ($):

```
$  BSTAR DCM
```

For information about BASEstar Classic commands, see the *BASEstar Classic Command Line Interface User's Guide*.

To use the BASEstar Classic menu system, enter the following command:

```
$  BSTAR/MENU
```

For information about the BASEstar Classic menu system, see the *BASEstar Classic Command Line Interface User's Guide*. For information about the BASEstar Classic callable services, refer to the *BASEstar Classic Application Programming Interface Reference Guide*.

3.2 Supported Functions

This section describes the functions that are supported by the DAS for OMNI software devices.

The DAS for OMNI software supports the following BASEstar Classic functions using MMS compliant devices:

- Read and Write Named and Unnamed Variables
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Receive Unsolicited Status (Status Indication)
- Receive Unsolicited Aborts and Concludes (Abort/Conclude Indication)
- Read Device Status and Identity
- Download and Upload Domains (Programs and Data)
- Start and Stop Programs (Program Invocations)

- Get Device Directory (Get Name List)

The DAS for OMNI software supports the following BASEstar Classic functions using SINEC-AP/H1 compliant devices:

- Read and Write Named and Unnamed Variables
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Receive Unsolicited Status (Status Indication)
- Receive Unsolicited Aborts and Concludes (Abort/Conclude Indication)
- Read Device Status and Identity
- Read, Write and Exchange SINEC-AP and SINEC-H1 Messages

The DAS for OMNI software supports the following BASEstar Classic functions using SINEC-S7 compliant devices:

- Read and Write Unnamed Variables
- Receive Unsolicited Variables (Information Reports, Read and Write Indications)
- Receive Unsolicited Status (Status Indication)
- Receive Unsolicited Aborts (Abort Indication)
- Read Device Status and Identity

3.2.1 Connecting To a Device

The MODIFY DEVICE/ENABLE command is used to set a device online. The DAS then attempts to connect to the device. If the connection is successful then the connection will remain active until either the device is disabled or an abort/conclude is generated. Should the initial connection request fail, further connection attempts will be made when the devices is requested to perform a function, ie. read status or if the reconnect interval defined for the device has expired.

NOTE: The DAS will not accept incoming connection requests from the remote VMD.

Example 3–1 shows an example of enabling a device.

The first time a device is enabled all the OMNI software related data structures have to be built up by the DAS, so this could take a few seconds or more if a lot of variables are defined.

Example 3–1 Connecting to an MMS/SINEC-S7/SINEC-AP/SINEC-H1 Device

```
DCM> modify device TEST_PLC /enable
DCM>
```

3.2.2 Disconnecting a Device

The MODIFY DEVICE/DISABLE command is used to set a device offline, Example 3–2 shows an example of this.

When a device disabled an abort of the connection is sent to the device.

Example 3–2 Disconnecting an MMS/SINEC-S7/SINEC-AP/SINEC-H1 Device

```
DCM> modify device TEST_PLC /disable
DCM>
```

3.2.3 Read and Write Data Functions

Use the BASEstar Classic read data and write data functions to read data from and write data to a specific address in device storage. Address syntax is common for all MMS compliant devices and allows access to both "Named" and "Unnamed" MMS variables. An MMS variable is a typed virtual object that represents an access path to a location on the device ("real" variable).

The Named Variable object describes access to the real variable using an application process determined name. It may be used to describe a computed variable or a real variable whose fixed address is not public.

The Unnamed Variable object describes access to the real variable using a device-specific address. It requires a known, fixed address for the real variable.

Table 3–1 shows the syntax for configuring named and unnamed variable objects.

Table 3–1 Supported Data Addressing

BASEstar Classic Data Address	OMNI Address
"XXX"	Named variable "XXX"
"N\XXX"	Named variable "XXX"
"U\YYY"	Unnamed variable at unconstrained address "YYY"
"S\YYY"	Unnamed variable at symbolic address "YYY"
"X\YYY"	Unnamed variable at numeric address "YYY"[1]
"M\YYY"	Message named "YYY"[2]
"MX\YYY"	Message named "YYY" - message exchange is performed[2].
"MW\YYY"	Message named "YYY" - message send is performed[2].

[1] "YYY" must be a decimal number.

[2] "Messages are supported in addresses for SINEC-AP only.

It is possible to access a variable defined on a domain. To access the named variable "ZZZ" defined on the domain "DDD" the DCM address "DDD.ZZZ" (or "N\DDD.ZZZ") must be used.

Note

When defining messages (M\ syntax) the "YYY" portion of the address must always be identical for a particular VMD since OMNI software only allows one message to be defined for a particular VMD. Note that the DAS always creates a remote VMD for each device and may also create a local VMD for that device. Message names for the local and remote VMD may be different.

The address of a physical point defined in DCM is used by the DAS as the name of the variable when defining the variable to OMNI software. The variable is created in OMNI software when the associated device is enabled. Subsequent variable accesses use the existing variable that was created.

Variables created in OMNI software are not deleted, so a variable remains defined to OMNI software until BASEstar Classic is shut down and restarted. For this reason, deleting a phypoint and creating it using a different format, but the same address, will result in erroneous and unpredictable results because the variable will be read using the original format and data will be returned to DCM using the modified format.

Table 3–2 shows supported data type formats for the DAS for OMNI software.

Table 3–2 Supported Data Types

DCM Data Type	OMNI Data Type	MMS Data Type
BIT	Boolean	Boolean
[S_]BYTE	Signed Integer 8	Signed Integer 8
[S_]WORD	Signed Integer 16	Signed Integer 16
[S_]LONGWORD	Signed Integer 32	Signed Integer 32
U_BYTE	Unsigned Integer 8	Unsigned Integer 8
U_WORD	Unsigned Integer 16	Unsigned Integer 16
U_LONGWORD	Unsigned Integer 32	Unsigned Integer 32
F_FLOATING	F-float	Floating Point
ARRAY[xx]:BIT	Bit String	Bit String
STRING:xx	Word Counted String	Octet String
ARRAY[xx]:format	Array	Array
STRUCTURE	Structure[1]	Structure

[1]The maximum number of fields that can be specified in a structure is limited. The exact number of fields allowed is determined by the size of the fields, so is not an exact number. The DAS will return ILAN$_PEINVFMT if the maximum number of fields has been exceeded.

In addition to the above supported mappings, additional mappings are possible by specifying a postfix operator on the BASEstar Classic physical point address. The postfix operator and the conversion performed is described in Table 3–3.

Table 3–3 Supported Address Postfix Values

Postfix Value	OMNI Data Type	MMS Data Type
\B	Boolean[1]	Boolean
\O	Word Counted String[2]	Octet String
\V	Null Terminated String[3]	Visible String
\A	Array[4]	Array

[1]This form allows a boolean array to be used rather than a bit string.

[2]BASEstar Classic word and longword values are byte swapped before the values are returned to the user.

[3]The BASEstar Classic data type must be STRING:xx. Visible Strings cannot be used as array elements.

[4]Normally a variable defined as a structure in BASEstar Classic must be matched by a variable defined as a structure in the remote VMD. If this qualifier is used, the BASEstar Classic structure is simplified as much as possible. If the structure can be simplified to an array, then an array data type is used when defining the variable to OMNI software.

Postfix operators are not allowed for message physical points since for messages, no interpretation of the data is done by the DAS.

Example 3–3 shows an example of the information displayed after issuing the READ DATA command.

Example 3–3 Read Data Screen

```
DCM> read phy X/dev = TEST_PLC

Point       : X
Device      : TEST_PLC
Address     : n\X
Format      : S_LONGWORD

Data:
    0       :              2

DCM>
```

Example 3–4 provides an example of the information displayed after issuing the READ DATA command when using data array.

Example 3–4 Read Data Screen with Data Array

```
DCM> read phy A/dev = TEST_PLC

Point       : A
Device      : TEST_PLC
Address     : n\A
Format      : ARRAY[3]:U_BYTE

Data:
    0       :              2
    1       :              4
    2       :              6

DCM>
```

Example 3–5 shows an example of the information displayed after issuing the WRITE DATA command.

Example 3–5 Write Data Screen

```
DCM> write phy X/dev = TEST_PLC

Point       : X
Device      : TEST_PLC
Address     : n\X
Format      : S_LONGWORD

Data value     0       :   2

DCM>
```

Example 3–6 shows an example of the information displayed after issuing the WRITE DATA command using a data array.

Example 3–6 Write Data Screen with Data Array

```
DCM> write phy A/dev = TEST_PLC

Point       : A
Device      : TEST_PLC
Address     : n\A
Format      : ARRAY[3]:U_BYTE

Data value     0       :   2
Data value     1       :   4
Data value     2       :   6

DCM>
```

3.2.4 Read and Write Message Functions Using ILAN$DEVICE_SPECIFIC

The SINEC-AP and SINEC-H1 message functions are implemented as BASEstar Classic device specific functions. SINEC-AP messages can also be used with physical points.

Table 3–4 lists ILAN$DEVICE_SPECIFIC function codes that are used to send and receive messages.

Table 3–4 ILAN$DEVICE_SPECIFIC Function Codes

Function Code	OMNI function	Description
5	omni_get_value	Read a message from the remote VMD
6	omni_put_value	Write a message to the remote VMD
7	omni_send_value	Write a message to the remote VMD
8	omni_exchange_data	Exchange a message with the remote VMD

Function code definitions are found in the file SYS$LIBRARY:DCM_OMNI$DEF.H.

To write, send or exchange a message the ILAN$DEVICE_SPECIFIC request buffer must be filled out according to the following syntax:

```
message-name:message-length.text
```

The message "text" will be written on the message "message-name" for the length "message-length".

To read a message the ILAN$DEVICE_SPECIFIC request buffer must be filled out according to the following syntax:

```
"message-name:message-length"
```

The message "text" will be read from the message "message-name" for the length "message-length" and copied into the response buffer. Example 3–7 shows a C program reading a message.

Example 3–7 Read message procedure

```
#include <descrip.h>
#include <ilandefs.h>
#include <dcm_omni$def.h>

main()

{

$DESCRIPTOR (dev_dsc,"OMNI_DAS_SRV");
char req_buf[100];
char res_buf[100];
short req_len = 100;
short res_len = 100;
short ret_len;
int func_code;

long status;

/* Read a message */

func_code = dcm_omni$k_read_message;
strcpy (req_buf,"OMNI_DAS_MSG:10");

memset (res_buf,0); /* clean up the buffer */

status = ILAN$DEVICE_SPECIFIC  (0,
                                &dev_dsc,
                                &func_code,
                                &req_len,
                                req_buf,
                                &res_len,
                                res_buf,
                                &ret_len);

if (status != ILAN$_NORMAL)
   {
   /* error recovery as you like */
   }

printf ("Received message: %s\n",res_buf);

}
```

Example 3–8 shows a C program writing a message.

Example 3–8 Write message procedure

```
#include <descrip.h>
#include <ilandefs.h>
#include <dcm_omni$def.h>

main()

{

$DESCRIPTOR (dev_dsc,"OMNI_DAS_SRV");
char req_buf[100];
char res_buf[100];
short req_len = 100;
short res_len = 100;
short ret_len;
int func_code;

long status;

/* Write a message */

func_code = dcm_omni$k_write_message;
strcpy (req_buf,"OMNI_DAS_MSG:10.message !!"); /* "message !!" */

status = ILAN$DEVICE_SPECIFIC  (0,
                                &dev_dsc,
                                &func_code,
                                &req_len,
                                req_buf,
                                &res_len,
                                res_buf,
                                &ret_len);

if (status != ILAN$_NORMAL)
   {
   /* ... */
   }

}
```

3.2.5 SINEC-AP Read and Write Message Functions Using Physical Points

When sending, receiving or exchanging a message, the first word of the buffer always contains the actual length of the data transferred. The maximum length of the message is the size of the physical point being defined. The DAS does no interpretation of the message that is transferred.

Only one message can be created for a particular VMD. However, multiple physical points can be created for a particular device as long as the name of the message is identical. Also keep in mind that both a local and remote VMD can be created for a single device, and that a unique message can be created for each. For all solicited access, the message is created on the remote VMD with the exception of a physical point created with a "MW" address. For all unsolicited access, the message is created on the local VMD.

3.2.5.1 Reading a message

To read a message from the remote VMD, an address of the form "M\YYY" must be specified where "YYY" is the name of the message.

3.2.5.2 Writing a message

To write a message to the remote VMD, an address of the form "M\YYY" or "MW\YYY" must be specified where "YYY" is the name of the message. If "M" is specified, then the message is defined on the remote VMD and an acknowledged write is performed. If "MW" is specified, then the message is defined on the local VMD and an unacknowledged write is performed.

Attempting to read from an address specified as "MW" results in an error being returned to the user.

3.2.5.3 Exchanging a message

To exchange a message with the remote VMD, an address of the form "MX\YYY" must be specified where "YYY" is the name of the message. To exchange a message, a write must first be performed. The write fills in a buffer in the DAS with the data to be exchanged with the VMD. When a read is performed the message is exchanged and the resulting message is returned to the user. If multiple reads are performed without an intervening write, the same data is written to the remote VMD.

3.2.6 Download, Upload and Directory Functions

The upload function transfers the contents of a device's memory to an OpenVMS file. The download function transfers the contents of an OpenVMS file to a device's memory. The directory function lists a device's memory.

In order to avoid errors, you will need to know how OpenVMS files are mapped onto the MMS device organization.

DCM file services provide functionalities to operate on remote device files: file download, file upload, file directory and device program management.

OMNI software supports the MMS File Management Services (primitives FGET, FPUT, FDIR, FRENAME and FDELETE), however they are seldom supported by any of the modern MMS devices. Therefore, these primitives are not supported by the DAS for OMNI software. In this manual, a file is always intended to be a file storing data or instructions related to a Domain, ie. programs, and parameters.

All operations on files are intended to operate on Domains and it is not possible, using the DAS for OMNI software to access the MMS file management primitives.

The download function transfers the content of the host Domain and Capability files to a Domain. The upload function transfers the content of a Domain to the OpenVMS Domain and Capability files.

The directory function lists the names of the Domains and the PIs found on a remote VMD.

Domain Restrictions

In general a Domain must first be uploaded from a device before it may be downloaded, this is due to vendor specific formatation of a devices Domain. Prior to a Domain being downloaded it must first be deleted at the remote MMS device, since the DAS for OMNI software does not support deleting of domains.

Example 3–9 shows an example of the information displayed after issuing the DIRECTORY command.

Example 3–9 Directory Screen

```
DCM> DIRECTORY GEF_PLC

Directory of device  GEF_PLC  at 17-MAY-1991 09:30:47.46
Path:  *.*

Domain: I_O_OVERRIDE_TABLE
Domain: I_O_TABLE
Domain: REGISTER_TABLE
Domain: SUBR_VECTOR_ADDRESSES
Domain: USER_LOGIC
PI    : DEMO_PI

Total of 6 files, size 0

DCM>
```

Example 3–10 shows an example of the use of the UPLOAD and DOWNLOAD commands.

Example 3–10 Upload and Download Screen

```
DCM> UPLOAD GEF_PLC USER_LOGIC.DOM/DEV=USER_LOGIC
DCM>
...
DCM> DOWNLOAD GEF_PLC USER_LOGIC.DOM/DEV=USER_LOGIC
DCM>
```

The standard extension for the content file is .DOM: you can use any extension for the content file, the capability file has always the same name with extension .CAP.

3.2.7 Start and Stop Functions

The start and stop functions change the status of a Program Invocation.

A Program Invocation can be reusable or not reusable. If a Program Invocation is reusable the start command will switch it from IDLE to RUNNING and the stop command will switch it from RUNNING to STOPPED and then to IDLE again.

If a Program Invocation is not reusable the start command will switch it from IDLE to RUNNING and the stop command will switch it from RUNNING to STOPPED and then to UNRUNNABLE.

To specify the Program Invocation name the OPTIONS field must be used with the following syntax:

```
DCM> START dev_name /START_OPTIONS="PI=pi_name"
```

and

```
DCM> STOP dev_name /STOP_OPTIONS="PI=pi_name"
```

A Program Invocation must exist on the device or created by the DAS for OMNI software. To create and delete program invocations the OPTION field must be used with the following syntax:

```
DCM> START dev_name /START_OPTIONS="PI=pi_name,CREATE=dom_name"
```

or

```
DCM> START dev_name /START_OPTIONS="PI=pi_name,CREATE=(dom_name_list)"
```

and

```
DCM> STOP dev_name /STOP_OPTIONS="PI=pi_name,DELETE"
```

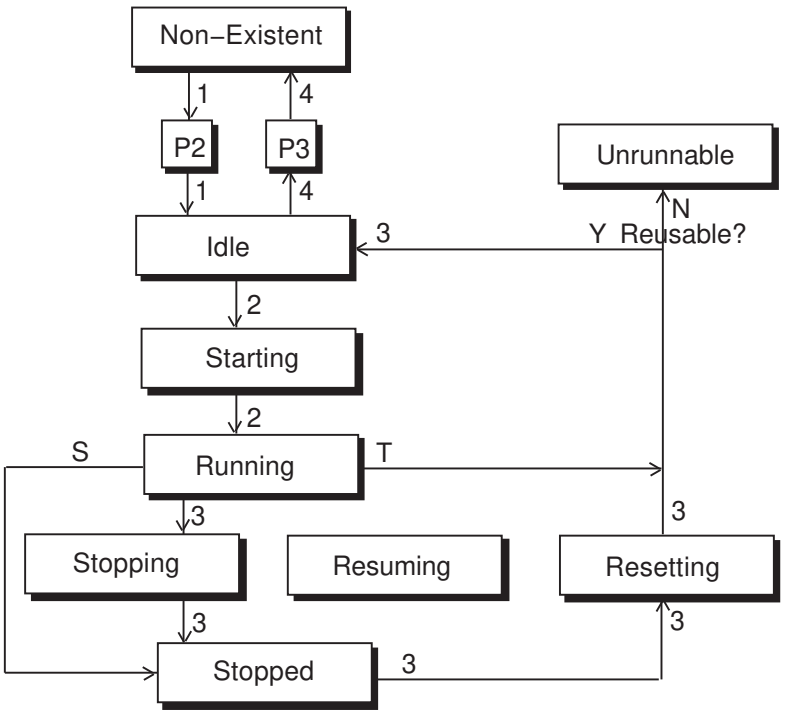
The Program Invocations created by the DAS for OMNI software are always reusable.

After a program terminates a START command can be entered again. After a program stop (error) a STOP command is needed before the program can be restarted.

If a Program Invocation is idle a STOP/DELETE program will work anyway.

Figure 3–1 shows the effects of the start and stop commands on a program invocation.

Figure 3–1 Program Invocations State TABLE



1: /START_OPTIONS="CREATE=..." option
2: START device command

T: program termination
S: program stop (error)

3: STOP device command
4: /STOP_OPTIONS="DELETE" option

Please note that many state transactions cannot be done through the DAS for OMNI software.

3.2.8 Read Status Function

The read status function issues a diagnostic status request to the device, interprets the device response, and returns the interpretation as a character buffer. An error message is displayed if a device definition does not match the device in the device response. If this occurs, you must correct the device definition before you can perform an upload function or download function for the device.

The *BASEstar Classic Command Line Interface User's Guide* gives detailed information about the READ STATUS command. To display the returned values for device status, enter the following at the BASEstar Classic prompt:

```
DCM> READ STATUS device-name/FULL
```

Example 3–11 shows an example of the information displayed after issuing the READ STATUS command.

Example 3–11 Read Status Screen

```
DCM> READ STATUS device-name/FULL

Status of device  GEF_PLC  at 17-MAY-1991 09:29:04.70

Vendor:               GE_Fanuc
Model:                SERIES_SIX_PC
Revision:             0120_0311_0312
Physical status:      0 (Operational)
Logical status:       0 (State Changes Allowed)
Local detail:         00000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000

DCM>
```

3.2.8.1 Logical Status Meaning

The logical status of an MMS device tells a remote user, the DAS for OMNI software, the MMS services that it is capable of supporting. A user of the DAS for OMNI software, may check this status to determine why a particular service, his request, is returning an error condition. The logical status code values and their meaning are described in Table 3–5.

Table 3–5 Logical Status Meaning

Logical Status Value	MMS Meaning
0	state changes allowed, all MMS services
1	no state changes allowed, Read Only services
2	limited services permitted, connect and status
3	supported services allowed, no PI services

3.2.8.2 Physical Status Meaning

The physical status of an MMS device tells a remote user, the DAS for OMNI software, the generic capabilities and operational state of the hardware. Further meaning is outside the scope of the MMS specification and may be determined from the vendors local status details. The physical status code values and their meaning are described in Table 3–6.

Table 3–6 Physical Status Meaning

Physical Status Value	MMS Meaning
0	Operational, ie. able to perform all tasks
1	Partially Operational, ie. not all tasks can be performed
2	Inoperable, ie. not operational
3	Needs Commisioning, ie. maintenance

3.3 Automatic Data Collection

Data can be collected automatically through either polled or unsolicited data collection. These options are discussed in the following paragrahs.

3.3.1 Unsolicited Data

Unsolicited data involves a device sending a physical point to BASEstar Classic services or reading a physical point from BASEstar Classic without it being previously requested. In both MMS and SINEC-S7/AP/H1 this function is described as being either an Information Report, a Write Indication a Read Indication, or a Message Exchange Indication (SINEC-AP only).

BASEstar Classic services can accept unsolicited values from any device as long as the device and physical point are defined with unsolicited information. Refer to the *BASEstar Classic Command Line Interface User's Guide* for more information concerning unsolicited data. Refer to the manufacturer's documentation for your specific device for more information on how it can send or request unsolicited data (Information Report, Write Indication or Read Indication).

3.3.1.1 Information Report

OMNI devices are capable of sending unsolicited data to OpenVMS as Information Reports. The DAS for OMNI software allows you to define BASEstar Classic physical points for collecting unsolicited data. If the device sends up an Information Report the DAS forwards the message to BASEstar Classic. The BASEstar Classic unsolicited ID must equal "I\".

Information reports are supported for named and unnamed variables and for SINCEC-AP messages.

3.3.1.2 Write Indication

OMNI devices are capable of sending unsolicited data to OpenVMS as Write Indications. The DAS for OMNI software allows you to define BASEstar Classic physical points for collecting unsolicited data. If the device sends up a Write Indication the DAS forwards the message to BASEstar Classic. The BASEstar Classic unsolicited ID must equal "W\".

The difference between an Information Report and a Write Indication is in where the variable is defined. For an Information Report, the variable is defined on the Remote VMD. The Information Report sends the value of a variable that exists in the remote VMD. For a Write Indication, the variable is defined on the Local VMD. The Write Indication sends a value to the Local VMD from the remote peer. For a Write Indication, the variable does not need to exist on the Remote VMD.

Write indications are supported for named and unnamed variables and for SINEC-AP messages.

3.3.1.3 Read Indication

OMNI devices are capable of requesting unsolicited data from OpenVMS as Read Indications. The DAS for OMNI software allows you to define BASEstar Classic physical points for responding to unsolicited read requests.

It is possible to configure the physical point such that replies are automatically returned to the device, or to configure it such that an explicit write request must be sent to the device to complete the read indication. These two options are discussed below.

Read indications are supported for named and unnamed variables and for SINEC-AP messages.

3.3.1.3.1 Automatic Reply to Read Indication To automatically send a reply, a BASEstar Classic logical point must be associated with the physical point. If the device sends a Read Indication the DAS reads the associated BASEstar Classic logical point and returns the result to the device. If the BASEstar Classic logical point does not exist or has a bad status, then an error is returned to the device. The BASEstar Classic unsolicited ID must equal "R\LOGICAL_POINT_NAME", where LOGICAL_POINT_NAME is the name of the logical point to read to get the value to return to the device. The size of the format of the logical point must match the size of the data defined by the variable being requested by the Remote VMD.

Note

The logical point name is compiled when the variable is defined and the compiled name is used when accessing the data point when a Read Indication is detected. If the data manager global section is recreated, then the logical point name needs to be recompiled. To force recompiling of the logical point name delete the BCC$SYSDATA:ILAN$_SECTION.DAT global section backing file when restarting BASEstar Classic software.

The user can also request that a second point be updated when a BASEstar Classic logical point is read through a Read Indication. This capability allows an application to be notified when the Remote VMD has read a value from the OpenVMS system. The second point must have the same address as the Read Indication point with the addition of a "T\" prefix on the address and an unsolicited ID of "\T". The format of this physical point must be a "BIT". When a Read Indication occurs for the corresponding Read Indication point, the DAS forwards a message to BASEstar Classic with a value of "1". "Trigger" physical points are used only by the DAS and cannot be used to read or write values to a device.

3.3.1.3.2 Manual Reply to Read Indication There may be cases where the user wants to be notified of a read indication, and then send the reply to the device. In this case a "Trigger" physical point must be set up as described above for the automatic case. A physical point must also be configured with an unsolicited ID of "\R". Note that in this case there is no BASEstar Classic logical point name configured in the unsolicited ID. When a read indication is detected by the DAS, the corresponding trigger point for that physical point is updated. The user is then responsible for sending a write request to the DAS for the corresponding physical point associated with the trigger physical point. No further read indications for that physical point will be accepted until the previous read indication has been completed by writing back down to the device.

3.3.1.4 Message Exchange Indication

SINEC-AP devices are capable of exchanging unsolicited data with OpenVMS as Message Exchange Indications. The DAS for OMNI software allows you to define BASEstar Classic physical points for exchanging unsolicited data.

It is possible to configure the physical point such that replies are automatically returned to the device, or to configure it such that an explicit write request must be sent to the device to complete a the message exchange indication. These two options are discussed below.

Message exchange indications are supported for SINEC-AP messages only.

3.3.1.4.1 Automatic Reply to Exchange Message Indication To automatically send a reply, a BASEstar Classic logical point must be associated with the physical point. If the device sends up a Message Exchange Indication the DAS reads the associated BASEstar Classic logical point and returns the result to the device. If the BASEstar Classic logical point does not exist or has a bad status, then an error is returned to the device. The BASEstar Classic unsolicited ID must equal "X\LOGICAL_POINT_NAME", where LOGICAL_POINT_NAME is the name of the logical point to read to get the value to return to the device. The size of the format of the logical point must match the size of the data defined by the message being requested by the Remote VMD. The data sent by the device is used to update the associated physical point.

Note

The logical point name is compiled when the message is defined and the compiled name is used when accessing the data point when a Message Exchange Indication is detected. If the data manager global section is recreated, then the logical point name needs to be recompiled. To force recompiling of the logical point name delete the BCC$SYSDATA:ILAN$_SECTION.DAT global section backing file when restarting BASEstar Classic software.

3.3.1.4.2 Manual Reply to Message Exchange Indication There may be cases where the user wants to examine the message sent in a message exchange indication, and then send the reply to the device. In this case the physical point must be configured with an unsolicited ID of "\X". Note that in this case there is no BASEstar Classic logical point name configured in the unsolicited ID. When a message exchange indication is detected by the DAS, the corresponding physical point is updated. The user is then responsible for sending a write request to the DAS for the physical point. No further message exchange indications for that physical point will be accepted until the previous messge exchange indication has been completed by writing back down to the device.

Table 3–7 summarizes unsolicited ID values and their meanings:

Table 3–7 Unsolicited ID Values

Unsolicited ID	Description
I\	Information Report
W\	Write Indication
R\LOGICAL_POINT_NAME	Read Indication (Automatic Reply)
R\	Read Indication (Manual Reply)[1]
X\LOGICAL_POINT_NAME	Message Exchange Indication (Automatic Reply)
X\	Message Exchange Indication (Manual Reply)
T\	Trigger

[1]Note that in this case, a corresponding "trigger" physical point must be configured.

3.3.1.5 Local and Remote VMDs

Information Report and solicited variables and messages are defined on the Remote VMD only. Read and Write Indication variables and messages are defined on the Local VMD only. A Remote VMD is created in OMNI software for each BASEstar Classic device. A Local VMD is created for each BASEstar Classic path definition. To force creation of a unique Local VMD for each Remote VMD, create a separate path for each device.

Example 3–12 shows an example of creating unsolicited physical point definitions.

Example 3–12 Unsolicited Point Definition

```
DCM> create phy U1/dev=TEST_PLC/addr="N\U1"/form=long/unsol="I\"
DCM> show phy U1/dev=TEST_PLC/FULL
Physical Point : U1
   Device                : TEST_PLC
   Description           :
   Address               : N\U1
   Data Format           : S_LONGWORD
   Write Enabled         : No
   Collection            : Unsolicited    ID : I\
   Collection Enabled    : No
   Distribution Destination
       Log               : No
       BASEstar Classic Data Mgt :
   Number of Connections :              0
   Defined on  ....
   Definition last updated on  ....

DCM>
```

(continued on next page)

Example 3–12 (Cont.) Unsolicited Point Definition

```
DCM> create phy U2/dev=TEST_PLC/addr="N\U2"/form=long/unsol="R\P1"
DCM> show phy U2/dev=TEST_PLC/FULL
Physical Point : U2
   Device                : TEST_PLC
   Description           :
   Address               : N\U2
   Data Format           : S_LONGWORD
   Write Enabled         : No
   Collection            : Unsolicited    ID : R\P1
   Collection Enabled    : No
   Distribution Destination
       Log               : No
       BASEstar Classic Data Mgt :
   Number of Connections :              0
   Defined on  ....
   Definition last updated on  ....

DCM>

DCM> create phy T2/dev=TEST_PLC/addr="T\N\U2"/form=BIT/unsol="T\"
DCM> show phy T2/dev=TEST_PLC/FULL
Physical Point : U2
   Device                : TEST_PLC
   Description           :
   Address               : T\N\U2
   Data Format           : S_LONGWORD
   Write Enabled         : No
   Collection            : Unsolicited    ID : T\
   Collection Enabled    : No
   Distribution Destination
       Log               : No
       BASEstar Classic Data Mgt :
   Number of Connections :              0
   Defined on  ....
   Definition last updated on  ....

DCM>

DCM> create phy M1/dev=TEST_PLC/addr="M\M1" -
_DCM> /form=struct"(array[10]:WORD)"/unsol="W\"
DCM> show phy M1/dev=TEST_PLC/FULL
Physical Point : M1
   Device                : TEST_PLC
   Description           :
   Address               : M\M1
   Data Format           : S_LONGWORD
   Write Enabled         : No
   Collection            : Unsolicited    ID : W\
   Collection Enabled    : No
   Distribution Destination
       Log               : No
       BASEstar Classic Data Mgt :
   Number of Connections :              0
   Defined on  ....
   Definition last updated on  ....

DCM>
```

3.3.2 Pollsets/Scattered Read

BASEstar Classic physical points may be grouped together into pollsets to optimize data collection and to provide the ability to read a group of points on a specified time interval.

DCM allows you to read groups of physical points (variables) using the scattered pollset feature. Any combination of physical points may be included in a pollset. The use of the /SCATTERED qualifier is mandatory when creating the pollset.

Figure 3–2 shows an example of creating a scattered pollset for reading groups of named variables.

Figure 3–2 Scattered Pollset Definition

```
DCM> create pollset POLL1/device=TEST_PLC/interval=15/scattered
DCM> create phypoint P1/dev=TEST_PLC/format=long/addr=VMD_INT32  -
_DCM> /poll=POLL1/COLLECT
DCM> create phypoint P2/dev=TEST_PLC/format=word/addr=OSAPWORD  -
_DCM> /poll=POLL1/COLLECT
DCM> SHOW POLL */DEV=TEST_PLC/FUL

Device Name : TEST_PLC
Pollset Name: POLL1
    Interval        :   15
    Enabled         :   No
    Format          :   SCATTERED
    Start Address   :   VMD_INT32
    Number elements :   2
    IO blocks       :   1
    Size (in bytes) :   6
    Buffers not dist:   0
    Defined on  ....
    Definition last updated on ....

Permanent Lka Ctr           :0      Temporary Lka Ctr           :0
Permanent Late Ctr          :0      Temporary Late Ctr          :0
Permanent Device Errors     :0      Temporary Device errors     :0
Permanent No Buffers errors :0      Temporary No Buffers errors :0
 BASEstar Classic point name:       DCM Phypoint name:
                                           P1
                                           P2
DCM>
```

This service provides the equivalent of calling the omni_group_variable service for each physical point (variable) in the pollset.

For more information on physical points, see the *BASEstar Classic Command Line Interface User's Guide*.

A

Returned Values and Associated Error Messages

This appendix contains returned values and associated error messages for the DAS for OMNI software.

The return values generated by DAS for OMNI software are either standard ILAN$_ values or are from OMNI software. The return values generated by OMNI have the facility code of OMNI_.

To enable message translation of these OMNI_ messages for each process, enter the following command:

```
$ SET MESSAGE SYS$MESSAGE:OMNI_CODES_MSG.EXE
```

For a full list of OMNI errors please refer to the *OMNI Application Interface User's Guide*. This is a short guide to the most commonly seen errors.

OMNI-E-INITFAIL, Initiate request failed,

Explanation: Error. An error occurred when attempting to connect to a device.

User Action: Check the device availability.

OMNI-E-CONNECTERR, Connect Error,

Explanation: Error. An error occurred when attempting to connect to a device.

User Action: Check local and remote ODS configurations. Make sure that you are using the supplied ILAN$SYSTEM_STARTUP.

OMNI-E-OBJ_UNDEFINED, Service Error Class- Definition: Object Undefined,

Explanation: Error. An error occurred when attempting to read or write to a physical point. In this case, the variable is not defined on the remote device.

User Action: Check that the variable is defined at the remote device.

OMNI-E-OBJ_ACCESS_DENIED, Service Error Class- Access: Denied,

Explanation: Error. An error occurred when attempting to read or write to a physical point. The device is not allowing access to the variable requested.

User Action: Check the variable address on the device.

OMNI-E-GETVAL_CNF_ERR, Get Value Confirm Error,

Explanation: Error. An error occurred when attempting to read a physical point.

User Action: Check the network and provider status to further diagnose the problem.

Returned Values and Associated Error Messages

OMNI-E-PUTVAL_CNF_ERR, Put Value Confirm Error,

Explanation: Error. An error occurred when attempting to write to a physical point.

User Action: Check the network and provider status to further diagnose the problem.

OMNI-E-OBJ_NOT_EXISTEN, Service Error Class- Access: Non-Existent,

Explanation: Error. An error occurred when doing an upload, download, start or stop function. You referred to a nonexistent Domain or PI.

User Action: Issue a DIRECTORY command to find valid Domains and PIs.

OMNI-E-STARTPI_REQ_ERR, Start Program Invocation Request Service Error,

Explanation: Error. An error occurred when attempting to do a "start" command. The /START_OPTIONS qualifier was probably specified improperly.

User Action: Specify the correct /START_OPTIONS qualifier in your start command.

OMNI-E-RESETPI_REQ_ERR, Reset Program Invocation Request Service Error,

Explanation: Error. An error occurred when trying to stop a PI. The /STOP_OPTIONS qualifier was probably specified improperly.

User Action: Specify the correct /STOP_OPTIONS qualifier on your stop command.

OMNI-E-OBJ_STATE_CONFL, Service Error Class- Service: Object State Conflict,

Explanation: Error. An error occurred when attempting to start or stop a PI. The start or stop was probably done twice.

User Action: None.

OMNI-F-INVFUNC, Connection is not in a valid state for this omni function,

Explanation: Fatal. The device or Network has aborted or concluded the connection.

User Action: Check physical and network connections to the device.

OMNI-E-GET-CAPABILITY, Get Capability List Confirm Error,

Explanation: Error. An error occurred when attempting to read device status.

User Action: Check that the device is still running.

OSAP-E-ASONOTCRE, Association not created,

Explanation: Error. An error occurred when attempting to connect to a device.

User Action: Check local and remote configurations.

B
Logged Messages

The messages in the following sections are logged to the BASEstar Classic history file by the DAS for OMNI software. These messages are logged to provide more detailed diagnostic information than what is supplied by the returned status values.

Messages logged to the history file for DAS for OMNI software use event class 21 and event type 32.

To view all messages logged by this DAS use the following syntax:

```
$ BSTAR
BSTAR> SHOW HISTORY/EVENT=21.32.*
```

OMNI software returns status results in an I/O status block consisting of three status fields, General, Provider and Network. The DAS logs each of these error statuses (if present) when logging status results.

B.1 NI Logged Messages

The following messages are logged by the Network Interface (NI).

Unable to initialize OMNI.

Explanation: Error. The call to omni_initialize failed.

User Action: Check the status returned and take action based on the status.

B.2 PE Logged Messages

The following messages are logged by the Protocol Emulator (PE).

Error calling omni_abort ( ).

Explanation: Error. The call to omni_abort failed.

User Action: Check the status returned and take action based on the status.

Device !AD connected.

Explanation: Informational. The DAS has successfully connected to the device.

User Action: None.

Device !AD connection aborted.

Explanation: Informational. The connection to the device has been successfully aborted.

User Action: None.

Device !AD association concluded by the presentation layer.

Explanation: Informational. The DAS has successfully accepted a request to conclude the association. The DAS submits a request to Device Connection Management to abort the connection and to transition the device to not connected and not reachable.

User Action: None.

Device !AD association concluded by the device.

Explanation: Informational. The DAS has received a conclude indication. The DAS submits a request to Device Connection Management to abort the connection and to transition the device to not connected and not reachable.

User Action: None.

Device !AD association aborted.

Explanation: Informational. The DAS has received an abort indication. The DAS submits a request to Device Connection Management to abort the connection and to transition the device to not connected and not reachable.

User Action: None.

Error creating local VMD.

Explanation: Error. An error occurred when attempting to create the local VMD using the OMNI API.

User Action: Examine the error message returned and take action based on the error found.

Error creating remote VMD.

Explanation: Error. An error occurred when attempting to create the remote VMD using the OMNI API.

User Action: Examine the error message returned and take action based on the error found.

Error calling omni_connect ( ).

Explanation: Error. An error occurred when attempting to connect to the remote VMD using the OMNI omni_connect service.

User Action: Examine the error message returned and take action based on the error found. Check local and remote ODS configurations. Check physical and network connections to the device.

Error creating simple OMNI type !AD for point !AD.

Explanation: Error. An error occurred when attempting to create the given data type. The DAS attempts to create named types for all "simple" (i.e. not a structure or array) data types.

User Action: Examine the error message returned and take action based on the error found.

Error creating structure OMNI type !AD for point !AD.

Explanation: Error. An error occurred when attempting to create the given data type. The data type name is formed from the fields in the structure so that the same structure definition can be used for multiple physical points.

User Action: Examine the error message returned and take action based on the error found.

Error creating array OMNI type !AD for point !AD.

Explanation: Error. An error occurred when attempting to create the given data type. The data type name is formed using the number of elements in the array so that the same array definition can be used for multiple physical points.

User Action: Examine the error message returned and take action based on the error found.

Error creating domain !AD for point !AD.

Explanation: Error. An error occurred when attempting to create a domain for a physical point that has a domain name embedded in it.

User Action: Examine the error message returned and take action based on the error found.

Error creating named variable for point !AD.

Explanation: Error. An error occurred when attempting to create a named variable for the physical point. The name of the named variable is taken from the address field of the physical point.

User Action: Examine the error message returned and take action based on the error found.

Error creating unnamed variable for point !AD.

Explanation: Error. An error occurred when attempting to create an unnamed variable for the physical point.

User Action: Examine the error message returned and take action based on the error found.

Error creating a message for point !AD.

Explanation: Error. (SINEC-AP devices only.) An error occurred when attempting to create a message for the physical point. The message object is used to transfer messages to and from the device.

User Action: Examine the error message returned and take action based on the error found.

Error compiling BASEstar logical point !AD.

Explanation: Error. An error occurred when calling BCC$COMPILE_ POINT_NAME for the given logical point. The logical point name is contained in the unsolicited ID of the physical point being configured for unsolicited data collection.

User Action: Examine the error message returned and take action based on the error found.

Error getting physical status attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_physical_status. This call is made to decode the results of an omni_get_remote_attributes call or of an unsolicited status indication.

User Action: Examine the error message returned and take action based on the error found.

Error getting logical status attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_logical_status. This call is made to decode the results of an omni_get_remote_attributes call or of an unsolicited status indication.

User Action: Examine the error message returned and take action based on the error found.

Error getting vendor name attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_vendor. This call is made to decode the results of an omni_get_remote_attributes call.

User Action: Examine the error message returned and take action based on the error found.

Error getting model name attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_model. This call is made to decode the results of an omni_get_remote_attributes call.

User Action: Examine the error message returned and take action based on the error found.

Error getting revision number attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_revision. This call is made to decode the results of an omni_get_remote_attributes call.

User Action: Examine the error message returned and take action based on the error found.

Error getting local detail attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_local_detail. This call is made to decode the results of an omni_get_remote_attributes call or of an unsolicited status indication.

User Action: Examine the error message returned and take action based on the error found.

Error calling omni_get_remote_attributes ( ).

Explanation: Error. An error occurred when calling omni_get_remote_ attributes. This routine is called when a "READ STATUS" command is done from the BASEstar Device Connection Management CLI.

User Action: Examine the error message returned and take action based on the error found.

Error writing value for point = !AD.

Explanation: Error. An error occurred when calling omni_put_value for the given physical point. (This error can also be logged when calling omni_ exchange_data or omni_send_value.) The most likely cause for this error is that the variable does not exist in the remote VMD or that the data types do not match.

User Action: Examine the error message returned and take action based on the error found.

Error reading value for point = !AD.

Explanation: Error. An error occurred when calling omni_get_value for the given physical point. (This error can also be logged when calling omni_ exchange_data.) The most likely cause for this error is that the variable does not exist in the remote VMD or that the data types do not match.

User Action: Examine the error message returned and take action based on the error found.

Read not allowed from unacknowledged write point !AD.

Explanation: Error. An attempt was made to read a physical point that has an address of "MW\YYY" where "YYY" is the name of the message. "MW" physical points are message physical points that can be written to only.

User Action: Redefine the physical point to allow both reads and writes or do not attempt to read from the physical point. If the physical point is connected to a logical point, use SHOW POINT/VALUE=LAST to show the last value written to the logical point. If the physical point is redefined, Device Connection Management may need to be restarted for the change to take effect.

Solicited read not allowed from unsolicited point !AD.

Explanation: Error. Solicited reads of unsolicited physical points are allowed only for write indication (W\) or information report indication (I\) physical points. Read indication physical points (R\) are defined on the local VMD and not on the remote VMD, so a read cannot be attempted since the variable does not exist.

User Action: Do not attempt to read from the physical point. If the physical point is connected to a logical point, use SHOW POINT/VALUE=LAST to show the last value written to the logical point.

Write not allowed, no indication pending for point !AD.

Explanation: Error. An attempt was made to write to an unsolicited physical point that does not have a read indication pending. Solicited writes of unsolicited physical points are allowed only for write indication (W\) or information report indication (I\) physical points. Read indication physical points (R\) are defined on the local VMD and not on the remote VMD, so a write cannot be attempted unless a read indication is pending since the variable does not exist on the remote VMD.

User Action: Wait for a read indication trigger to occur before writing to the physical point.

Buffer too small for the message for point !AD.

Explanation: Error. The length of the message (as specified in the first word of the message) is too large for the buffer that has been allocated for it. The size of the buffer is the size of the physical point that is defined for the message.

User Action: Modify the first word of the message so that the size is less than or equal to the buffer that has been allocated.

Write required to message exchange point !AD before reading.

Explanation: Error. An attempt was made to read from a message exchange physical point before writing to it. Reading from a message exchange physical point is what triggers the message exchange, but if the physical point hasn't been written to, there is no data in the buffer to exchange with the device.

User Action: Write to the exchange physical point before attempting to read from it.

Error getting PI name attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_name. omni_get_attribute is called after successfully calling omni_get_remote_attributes using a class of omni_c_cls_pi.

User Action: Examine the error message returned and take action based on the error found.

Error getting Domain name attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_name. omni_get_attribute is called after successfully calling omni_get_remote_attributes using a class of omni_c_cls_dom.

User Action: Examine the error message returned and take action based on the error found.

Error getting Named Variable name attribute.

Explanation: Error. An error occurred when calling omni_get_attribute with an attribute of omni_c_attr_name. omni_get_attribute is called after successfully calling omni_get_remote_attributes using a class of omni_c_cls_named_var.

User Action: Examine the error message returned and take action based on the error found.

Error downloading file.

Explanation: Error. An error occurred when calling omni_download.

User Action: Examine the error message returned and take action based on the error found.

Error uploading file.

Explanation: Error. An error occurred when calling omni_upload.

User Action: Examine the error message returned and take action based on the error found.

A domain must be specified in the /DEVICE_FILE qualifier.

Explanation: Error. When attempting to do an upload or download, no domain name was specified in the /DEVICE_FILE=XXX qualifier.

User Action: Specify the domain name you would like to upload or download in the /DEVICE_FILE qualifier of the upload or download command.

Error creating domain !AD for file !AD.

Explanation: Error. An error occurred when creating the domain specified in the /DEVICE_FILE qualifier of the upload or download command.

User Action: Examine the error message returned and take action based on the error found.

PI start/reset/delete error for PI !AD.

Explanation: Error. An error occurred when attempting to start or stop (reset/delete) the given program invocation (PI).

User Action: Examine the error message returned and take action based on the error found.

Start/stop option !AD syntax error.

Explanation: Error. The option specified in the START or STOP command was not specified correctly or is not supported.

User Action: Check the DAS documentation for valid options and syntax for the START or STOP command.

Error calling omni_get_value ( ) for pollset !AD.

Explanation: Error. An error occurred when calling omni_get_value for a group of variables. The DAS uses omni_group_variables to group members of a pollset.

User Action: Examine the error message returned and take action based on the error found.

Polling collision - pollset !AD in use.

Explanation: Error. A new poll request has occurred before the previous one has completed.

User Action: Increase the polling interval of the pollset.

No variable handle for point !AD in pollset !AD.

Explanation: Error. A variable handle was not found for the given physical point. The variable handle is created as part of enabling the device.

User Action: If the problem persists, disable and reenable the device.

Error calling omni_group_variables ( ) for point !AD in pollset !AD.

Explanation: Error. An error occurred when attempting to group variables in a scattered pollset.

User Action: Examine the error message returned and take action based on the error found.

Device !AD!/Physical status: !AD!/Logical status: !AD.

Explanation: Informational. The device has delivered an unsolicited status indication to the DAS. The DAS logs the status to the history file.

User Action: None.

Local detail: !AD.

Explanation: Informational. The device has delivered an unsolicited status indication to the DAS. The DAS logs (as a bit string), the information in the local detail portion of the status.

User Action: None.

!AD indication received.

Explanation: Error. An error occurred when receiving an indication from the device. The type of indication received is logged.

User Action: Examine the error message returned and take action based on the error found.

!AD indication received. No unsolicited point found.

Explanation: Error. An indication was received for a variable that is not defined. The indication is rejected.

User Action: Determine the variable that the indication is for (by examining programmable controller logic) and create an unsolicited physical point that matches the variable that is being requested.

!AD ind rejected. Error calling BCC$GET_VALUE ( ), logical point = !AD.

Explanation: Error. A read indication was received and a matching physical point was found, but the call to get the current value of the logical point failed. The indication is rejected.

User Action: Examine the error message returned and take action based on the error found. If the logical point exists then it may be necessary to force recompiling of the logical point name by stopping BASEstar software, deleting the ILAN$_SECTION.DAT file and restarting BASEstar software.

!AD ind rejected. Error in point status for logical point !AD.

Explanation: Error. A read indication was received, a matching physical point was found and the call to BCC$GET_VALUE succeeded, but the status of the logical point is bad. The indication is rejected.

User Action: Examine the error message returned and take action based on the error found.

Read ind rejected. Previous read ind pending for point !AD.

Explanation: Error. A logical point was not specified from which to get the data to return to the requestor, so the DAS is waiting for a write request to the physical point (after generating a trigger update) and the write had not yet occurred when another read indication was received. The indication is rejected.

User Action: Write to the given physical point to satisfy the previous read indication.

Read ind rejected. No trigger point and no logical point for point !AD.

Explanation: Error. A read indication was received and a matching variable was found, but the physical point did not specify a logical point and a matching trigger physical point was not found. The indication is rejected.

User Action: Modify the physical point to add a logical point name in the unsolicited ID, if automatic replies to the read indication is desired, or create a matching trigger physical point, if automatic notification of the indication is desired, but a manual response to the read indication is required.

Unexpected !AD indication received.

Explanation: Error. An indication type was received that the DAS is not designed to process. The indication is ignored, but since the indication is not responded to, virtual memory may be consumed and not properly released.

User Action: Modify the programming in the programmable controller to stop sending this type of indication.

Index

A

Accessing DAS functions, 3–1
Additional software
  required, 2–2
Application Simple Name, 2–12
Automatic data collection, 3–13
  pollsets, 3–18
  unsolicited data, 3–13

C

Configuration
  NCL, 2–16
  ODS, 2–13
Configuration file
  device records, 2–13
  path records, 2–12
  type records, 2–11
Configuration File
  editing, 2–10
Configuring parameters
  ILAN$MAX_SPT_REQUESTS, 2–19

D

DAS
  See Device Access Software
Data types, 3–4
Defining DAS Constants, 2–19
Device Access Software
  accessing functions, 3–1
  defining constants, 2–19
  description of, 1–1
  functions, 1–3, 1–4, 1–5
  installation of, 2–1
  installation requirements, 2–1
  MMS functions, 3–1
  overview, 1–1
  SINEC-AP functions, 3–2
  SINEC-H1 functions, 3–2
  SINEC-S7 functions, 3–2
  supported devices, 3–1
  supported functions, 1–2, 3–1
  using, 3–1
Device records, 2–13
Devices
  communications, 1–1
Directory function, 3–9
Disk space required, 2–2
Download function, 3–9

E

Exchange message function, 3–6, 3–8
Exchange message indication
  automatic reply, 3–15

F

Files
  created during installation, 2–8
Functions, 3–1
  device specific, 3–6, 3–8
  directory, 3–9
  download, 3–9
  exchange message, 3–6, 3–8
  read data, 3–3
  read message, 3–6, 3–8
  read status, 3–12
  start, 3–10
  stop, 3–10
  upload, 3–9
  write data, 3–3
  write message, 3–6, 3–8

G

Group reads, 3–18

H

Hardware
  required, 2–1

I

Information report, 3–13
Installation, 2–1
  files created, 2–8
  messages, 2–8
  post installation tasks, 2–10
  procedure, 2–3

Installation (cont'd)
requirements, 2–1

L

Local VMD, 3–16

M

Message exchange indication, 3–15
manual reply, 3–15
Messages
error and return values, A–1
installation, 2–8
logged, B–1

N

NCL
configuration, 2–16
null internet, 2–16
RFC-1006, 2–17
transport templates, 2–16

O

ODS
attributes, 2–15
configuration, 2–13
OSI Tracing, 2–20

P

Path records, 2–12
Pollsets
scattered reads, 3–18
Post installation tasks, 2–10
Problem reporting, 2–20
Product failure, 2–20
Protocol Profile, 2–11

R

Read data function, 3–3
Read indication, 3–14
automatic reply, 3–14
manual reply, 3–14
Read message function, 3–6, 3–8
Read status function, 3–12
Remote VMD, 3–16

S

Scattered read
pollsets, 3–18
Shop floor equipment
setting up, 2–20
Sinec-AP
message support, 3–8
Sinec-AP and SINEC-H1
specific functions, 3–6
Software
required, 2–1
Specific functions
device, 3–6
SPT block usage, 2–19
Start function, 3–10
Stop function, 3–10
SYSGEN parameters, 2–18

T

Tracing
OSI messages, 2–20
Tuning ILAN$DEVSRV, 2–18
Type records, 2–11

U

Unsolicited data, 3–13
information report, 3–13
message exchange indication, 3–15
read indication, 3–14
write indication, 3–13
Upload function, 3–9

W

Write data function, 3–3
Write indication, 3–13
Write message function, 3–6, 3–8