BASEstar Open Device Connectivity

DASware User Guide

Order Number: AA-QT1HB-TE

April 1997

This document describes DASware, the Device Access Software (DAS) that allow BASEstar Open users and DEComni API applications to exchange data with partner applications in a protocol-independent way through the most commonly-used transport networks. This manual also describes the procedures to install and use DASware in the BASEstar Open and DEComni API environments.

Revision/Update Information:	This is a new document for the current release.
Operating System and Version:	BASEstar Open is available on a broad range of both hardware and software platforms. Refer to your *Software Product Description* for precise information.
Software Version:	Version 3.1

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, February 1996
Revised, September 1996
Revised, April 1997

© Digital Equipment Corporation 1997.

All Rights Reserved.

Possession, use, or copying of the software described in this documentation is authorized only pursuant to a valid written license from DIGITAL or an authorized sublicensor.

Digital Equipment Corporation makes no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation: Alpha AXP, BASEstar Open, DEC, DECmessageQ, DECnet, DECnet-DOS, DECosap, DEComni, DIGITAL, DIGITAL UNIX, FMS, LN03, MicroVAX, NAS, OpenVMS, OpenVMS Alpha, PATHWORKS, PDAS, Rdb/VMS, ReGIS, ThinWire, TK, ULTRIX, VAX, VAXcluster, VAX COBOL, VAX FORTRAN, VAX Pascal, VAX RMS, VMS/ULTRIX Connection, VT, and the DIGITAL logo.

The following are third-party trademarks:

MS, Microsoft, and MS-DOS are registered trademarks of Microsoft Corporation.
UNIX is a registered trademark licensed exclusively by X/Open Company Ltd.
Windows and Windows NT are trademarks of Microsoft Corporation.

All other trademarks and registered trademarks are the property of their respective holders.

Table of Contents

Preface..... vii

1 Introduction 1–1

What is DASware? 1–1
Transport Networks and Platforms Supported by DASware..... 1–2
What's Next?..... 1–2

2 Installing DASware on Your Platform 2–1

Installation on DIGITAL UNIX Systems 2–2
Preparing for DASware Installation 2–2
Installing DASware 2–4
Deinstalling DASware 2–4
Files Installed on Your System..... 2–5
Getting Help and Reporting Problems 2–5
Installation on OpenVMS Systems..... 2–7
Preparing for DASware Installation 2–7
Files Installed on Your System..... 2–8
Getting Help and Reporting Problems 2–8
Installation on Windows NT Systems..... 2–10
Preparing for DASware Installation 2–10
Installing DASware 2–10
Deinstalling DASware 2–11
Getting Help and Reporting Problems 2–11
Installation on HP-UX Systems..... 2–12
Preparing for DASware Installation 2–12
Files Installed on Your System..... 2–13
Getting Help and Reporting Problems 2–13

PART I Using BASEstar Open to Exchange Data Via Transport Networks

3 Overview..... 3–1

BASEstar Open Environment Components..... 3–2
Exchanging Data Through a Transport Network..... 3–4
Sending Data 3–4
Receiving Data 3–5
Encoding and Decoding Transport Network Data 3–7

4 Configuring BASEstar Open 4–1

Registering ODS Entries 4–2
Creating Application Services Objects 4–2
Creating Device Services Objects 4–2
Creating Protocol_Profiles 4–2
Creating VMDs 4–3
Creating Variables 4–4
Creating Data Services Objects 4–5
Creating Device_Data_Points for Receive Operations 4–5
Creating Device_Data_Points for Send Operations 4–5

5 A Complete Example 5–1

Example Overview 5–2
Before Running the Example 5–2
CLI Source 5–3
Exchanging Data Through the Transport Network 5–4

PART II Using DEComni API to Exchange Data Via Transport Networks

6 Overview 6–1

DEComni API Environment Components 6–2
Exchanging Data Through the Transport Network 6–4
Writing Data Through the Transport Network 6–4
Receiving Data from the Transport Network 6–5
Encoding and Decoding Transport Network Data 6–7

7 Configuring DEComni API 7–1

Registering ODS Entries 7–2
Setting the Application Profile 7–2
Creating VMD Definitions 7–2
Creating Variable Definitions 7–2

8 Programming 8–1

DEComni API Procedures for Data Exchange 8–2
Opening and Closing a Transport Network Connection 8–2
Sending Data Through the Transport Network 8–2
Receiving Data from the Transport Network 8–2

PART III Appendices

A ODS Entry Attributes for DASware A–1

Creating ODS Entries for the RS 232 Transport Network A–1
Through Direct Connection or LAT A–1
Through Telnet A–3
Creating ODS Entries for the TCP/IP Transport Network A–4
Creating ODS Entries for the OSI Transport Network A–5

B Configuring DASware OSI Transport Services on an HP-UX System.....B–1

Configuring OTS/9000 Software.....B–1
Obtaining Ethernet Card Information B–1
Configuring the OSI Transport for an 802.3 LAN (Null Internet Option).....B–2
Configuring the OSI Transport for an 802.3 LAN (Full Internet Option) B–4
Configuring the OSI Transport for Running the IVP.....B–5

Figures

Figure 1-1: Communicating Via a Transport Network 1–1
Figure 3-1: BASEstar Open Environment Components..... 3–2
Figure 3-2: BASEstar Open Objects and Operations 3–4
Figure 6-1: DEComni API Environment Components..... 6–2
Figure 6-2: DEComni API Definitions and Procedures 6–4

Tables

Table 1-1: Available Platforms and DASware Modules..... 1–2
Table 2-1: DASware Subsets (DIGITAL UNIX Platform)..... 2–2
Table 2-2: Pre-requisite Software for DIGITAL UNIX (BASEstar Open Environment)..... 2–3
Table 2-3: Pre-requisite Software for DIGITAL UNIX (DEComni API Environment) 2–3
Table 2-4: Transport Network Software (DIGITAL UNIX Platform)..... 2–3
Table 2-5: Pre-requisite Software for OpenVMS (BASEstar Open Environment)..... 2–7
Table 2-6: Transport Network Software (OpenVMS Platform)..... 2–7
Table 2-7: Pre-requisite Software for Windows NT (BASEstar Open Environment).....2–10
Table 2-8: Transport Network Software (Windows NT Platform).....2–10
Table 2-9: Transport Network Software (HP-UX Platform).....2–12
Table 3-1: Protocol-Specific Information Added to Outgoing Encoded Data (BASEstar Open) 3–5
Table 3-2: Protocol-Specific Information Removed from Incoming Data (BASEstar Open)..... 3–6
Table 3-3: Coding and Decoding Data by Datatype (BASEstar Open)..... 3–7
Table 4-1: Protocol_Profile Attributes (BASEstar Open)..... 4–2
Table 4-2: VMD Attributes (BASEstar Open)..... 4–3
Table 4-3: Device Services Reception Variable Attributes (BASEstar Open)..... 4–4
Table 4-4: Device Services Transmission Variable Attributes 4–4
Table 4-5: Device_Data_Point Attributes for Receive Operations (BASEstar Open)..... 4–5
Table 4-6: Device_Data_Point Attributes for Send Operations 4–5
Table 6-1: Protocol-Specific Information Added to Outgoing Encoded Data (DEComni API) 6–5
Table 6-2: Protocol-Specific Information Removed from Incoming Data (DEComni API)..... 6–6
Table 6-3: Coding/Decoding Data According to MMS Types (DEComni API)..... 6–7
Table 7-1: DEComni API Application Profiles for DAS Modules 7–2
Table 7-2: VMD Attributes (DEComni API)..... 7–2
Table 7-3: Variable Attributes (DEComni API) 7–3
Table 8-1: DEComni API Procedures for Data Exchange..... 8–2

Examples

Example 4–1: Creating Protocol_Profile Objects (BASEstar Open)..... 4–3
Example 4–2: Creating VMD Objects (BASEstar Open)..... 4–3
Example 5–1: Creating BASEstar Open Objects 5–3
Example 5–2: Exchanging Data Through the Transport Network 5–4
Example A–1: ODS Entries for RS 232 Direct Connection and LAT A–2
Example A–2: ODS Entries for RS 232 Telnet.....A–3
Example A–3: ODS Entries for TCP/IP A–4

Example A–4: ODS Entries for the OSI Transport Network.....A–5

Preface

This manual describes how to install and use DASware, the DIGITAL Device Access Software (DAS) that allows BASEstar Open users and DEComni API applications to exchange data with partner applications in a protocol-independent way through the most commonly used transport networks.

Multiplatform Applicability

Most of the information in this manual applies regardless of the operating system platform on which the DASware software is installed. Information relating to installation is typically platform-dependent.

Note:

Throughout this document, all references to UNIX systems apply collectively to both DIGITAL UNIX and HP-UX.

Intended Audience

This manual is addressed to:

- System managers who have to install and maintain DASware on any of the supported platforms.
- Application developers who have to configure and use BASEstar Open, or the DEComni API (or both) to exchange data through the transport networks supported by the DASware modules.

Structure of this Document

This document is organized as follows:

- Chapter 1 introduces you to DASware; Chapter 2 contains the installation procedures.
- Part I describes the environment components that are involved when DASware is accessed using BASEstar Open. It also describes how you can exchange data through the supported transport networks, and how you must configure BASEstar Open. A complete example is also provided.
- Part II describes the environment components that are involved when DASware is accessed using the DEComni API. It also describes how you can exchange data through the supported transport networks, and how you must configure the DEComni API.
- Part III contains the appendices.

For More Information

If you are using BASEstar Open services you will find related information in the following documents:

- *BASEstar Open Introduction*
- *BASEstar Open Reference Guide*
- *BASEstar Open Command Language Interface*
- *BASEstar Open Application Programming Interface*
- *BASEstar Open Messages*
- Platform-specific management guides

If you are using DEComni API, you will find related information in the following documents:

- *DEComni API and DEComni MMS User Guide*
- *DEComni API Guide to Using Omni Directory Services*
- *DEComni API Omni Definition Facility* User Guide (for supported platforms)
- *DEComni API Guide to Using OmniView* (for supported platforms)

1
Introduction

This chapter provides you with an overview of DASware and of its main features.

What is DASware?

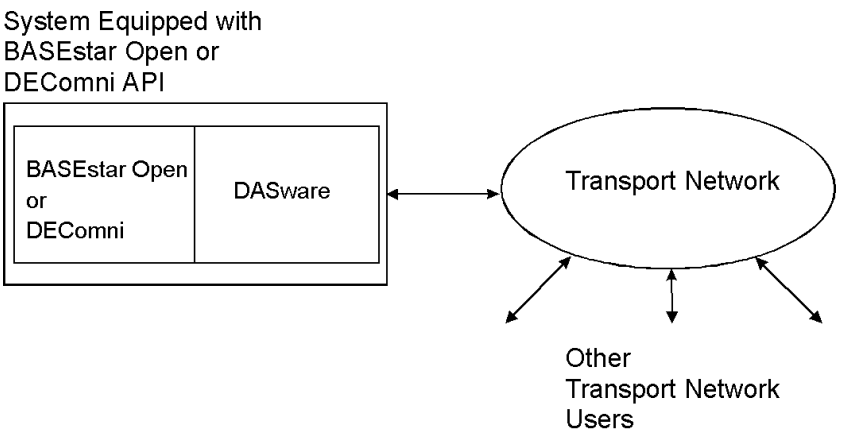
DASware is a DIGITAL product that allows both BASEstar Open users and DEComni API applications to exchange data with partner applications and devices in a protocol-independent way, through any of the supported transport networks.

Both BASEstar Open and the DEComni API offer generic device and network access, which is independent of the specific operations and protocols. This is possible because, for each supported protocol or device, there is a **Device Access Software (DAS)** module that matches device- and protocol-specific characteristics into standardized BASEstar Open services or DEComni API calls.

DASware is a set of modules that allow both BASEstar Open users and DEComni API applications to exchange data with partner applications and devices in a protocol-independent way, through the following transport networks:

- RS 232
- TCP/IP
- OSI Transport (ISO 8072)

Figure 1-1: Communicating Via a Transport Network



BASEstar Open users can send and receive data through the supported transport networks, simply by reading and writing the values of Device_Data_Points.

Similarly, DEComni applications use DEComni API procedures to exchange data. However, DEComni API is only supported on OpenVMS and DIGITAL UNIX platforms.

Transport Networks and Platforms Supported by DASware

Table 1-1 lists the DASware modules available on different platforms, which DAS modules are available and thus, which transport network can be used to exchange data through.

Table 1-1: Available Platforms and DASware Modules

Platform	Available DASware Modules
OpenVMS	RS 232, TCP/IP, and OSI Transport Layer (ISO 8072)
UNIX	RS 232, TCP/IP, and OSI Transport Layer (ISO 8072)
Windows NT	RS 232, and TCP/IP

When installing DASware on a given platform, a DASware module is installed for each of the supported transport networks.

What's Next?

In this chapter you have learnt about the main features of DASware.

If you intend to use DASware through BASEstar Open services, first read Chapter 2 to know how to install DASware on your system, and then read Part I of this manual to know how you must configure BASEstar Open and use its services to exchange data through the desired transport network(s).

If you intend to use DASware through the DEComni API, first read Chapter 2 to know how to install DASware on your system, and then read Part II of this manual to know how you must configure the DEComni API and use the API procedures to exchange data through the desired transport network(s).

2
Installing DASware on Your Platform

This chapter specifies the requirements and the procedures for installing DASware on the following platforms:

- DIGITAL UNIX
- OpenVMS
- Windows NT
- HP-UX

For each platform, a list is also provided of the packages and software components that you must install in order to make available the desired transport network(s) on your system.

This chapter provides the instructions for installing DASware in the BASEstar Open and DEComni API environments.

Installation on DIGITAL UNIX Systems

On DIGITAL UNIX systems, DASware is installed optionally as part of the DEComni API V3.1 installation procedure. You can either install DEComni API V3.1 as a separate product, or install it during the BASEstar Open V3.1installation procedure.

Preparing for DASware Installation

Before starting the installation procedure you should complete the preparation requirements outlined in this section. Requirements vary according to whether you are installing DASware for use via BASEstar Open services or via the DEComni API.

Table 2-1 lists the subsets included in the DASware software product. Note that a subset is provided for each of the transport networks supported on the DIGITAL UNIX platform.

Table 2-1: DASware Subsets (DIGITAL UNIX Platform)

DASware Module	Subset Name
RS232	DASTK232310
TCP/IP	DASTKTCP310
OSI Transport	DASTKOSI310

License Registration

DASware includes support for the DIGITAL UNIX License Management Facility (LMF). A License Product Authorization Key (License PAK) must be registered in the License Database (LDB) in order to use DASware on a newly-licensed node. The License PAK may be shipped along with the kit if you ordered the license and media together; otherwise, it is shipped separately to a location based on your license order.

If you are installing DASware as an update on a node already licensed for this software, you have already completed the License PAK registration requirements.

If you are installing prerequisite or optional software along with DASware, review the PAK status and install the PAKs for any prerequisite or optional software before you install DASware.

To register a license under the DIGITAL UNIX system, first log in as superuser.

You then have a choice of two ways to perform the PAK registration in the License Database (LDB):

- Before installing DASware

  At the superuser prompt, edit an empty PAK template with the `lmf register` command and include all the information on your License PAK as follows:

  ```
  # lmf register
  ```

- After installing DASware

  At the superuser prompt, edit the partially completed PAK template in `/usr/var/adm/lmf/`*`template`* with the `lmf register` command to add your unique License PAK information as follows:

  ```
  # lmf register - < /usr/var/adm/lmf/DASWARE-RUN
  ```

  After you register your license, use the following `lmf reset` command to copy the license details from the License Database (LDB) to the kernel cache:

  ```
  # lmf reset
  ```

For complete information on using the DIGITAL UNIX License Management Facility, see the *DIGITAL UNIX Guide to Software Licensing* or the `lmf(8)` reference page.

Software Requirements (BASEstar Open Environment)

If you are installing DASware in the BASEstar Open environment, check that the subsets listed in Table 2-2 have been installed on your system.

Table 2-2: Pre-requisite Software for DIGITAL UNIX (BASEstar Open Environment)

Subset Name and Description	Comments
BSTR310 BASEstar Open Server for DIGITAL UNIX Version 3.1	Ensure that the BASEstar Open Device Connectivity (DEComni) has been installed. In particular, check for the presence of the following subsets: • DOUBASE310 • DASTK232310 • DASTKRUN310 See the *BASEstar Open Server Inst. & Management Guide* for your platform for details.

Software Requirements (DEComni API Environment)

Check that the software products listed in Table 2-3 have been installed on your system.

Table 2-3: Pre-requisite Software for DIGITAL UNIX (DEComni API Environment)

Subset Name and Description	Comments
DOUBASE310 DEComni API for DIGITAL UNIX Version 3.1	In particular, check for the presence of the following subsets: • DASTK232310 • DASTKRUN310 See the *DEComni API Installation Guide* for your platform for details.

Transport Network Software

You must also ensure that the appropriate transport network software have been installed and configured. Table 2-4 specifies the necessary transport network software for each supported transport network.

Table 2-4: Transport Network Software (DIGITAL UNIX Platform)

Transport Network	Transport Network Software
RS232	The OSFLAT subset is required if LAT is used. The OSFLAT and OSFNET subsets are required if Telnet is used (the capability for communicating via serial ports is embedded in the operating system).
TCP/IP	The OSFCLINET and OSFINET subsets are required.
OSI Transport	The DNABASE and DNAPGMR subsets are required.

Disk Space Requirements

The table that follows lists the disk space required to install and run each DASware module.

Usage (Kbytes)	(root)	/usr/opt	/var/opt
Installation	0	8000	2000
Permanent	0	8000	2000

Backing Up Your System Disk

DIGITAL recommends that you back up your system disk before installing any software. For details of how to perform a system disk backup, refer to your DIGITAL UNIX documentation.

Installing DASware

Installing DASware and running the Installation and Verification Procedure (IVP) on your DIGITAL UNIX system takes approximately 5 to 10 minutes.

Install DASware as follows:

1. Log onto the system as superuser (root).
2. Mount the distribution media on the desired location (for example, mount `-dr /dev/rz4c /CDROM`).
3. Change your working directory to the kit location (for example, `cd /CDROM/DASTKRUN310`).
4. Issue the *`setld -l`* command to load the product onto the system, and follow the installation dialogs.

Running the IVP

During the installation procedure, you can run the IVP as part of the installation. If you do not run the IVP at that time, or if the product does not work correctly, you can run the IVP at any time by executing the following command:

```
setld -v DASTKRUN310
```

Deinstalling DASware

Deinstall DASware as follows:

1. Log onto the system as superuser (root).
2. Change your working directory to the kit location (for example, `cd /CDROM/DASTKRUN310`).
3. Issue the *`setld -d`* command:

```
setld -d DASTKRUN310
setld -d DASTK232310
setld -d DASTKTCP310
setld -d DASTKOSI310
```

Files Installed on Your System

This section lists the directories and files created by DASware.

/usr/opt/dastk_310/include:

omni_integrator16_defs_include.h

/usr/opt/dastk_310/lib:

dastk_232init.o

dastk_tcpinit.o

libdastk.a

libdastkosi.a

dastk_osiinit.o

dastksts.cat

libdastk232.a

libdastktcp.a

libdastkx25.a

/usr/opt/dastk_310/shlib:

libdastk.so

libdastk232.so

libdastkosi.so

libdastktcp.so

libdastkx25.so

/var/opt/dastk_310/examples:

dastk_ivp.c

dastk_ivp_clp

runivp_dastk

Getting Help and Reporting Problems

If an error occurs while you are using DASware, and you believe the error is the result of a problem associated with the product, take one of the following actions:

- If you have a basic or DECsupport Software Agreement, call your Customer Support Center (CSC). The CSC provides telephone support for high-level advisory and remedial assistance.
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you purchased the product within the last 90 days and you think the problem is casued by a software error, you can submit an SPR.

If you submit an SPR, please 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. 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, relevant data files and so forth.

4. Provide a list of the program.
5. If the program is longer than 50 lines, submit a copy of it on machine-readable media (floppy diskette or magnetic tape). If necessary, also submit a copy of the program library used to build the application.
6. Report only one program per SPR. This will facilitate a faster response.
7. Mail the SPR package to DIGITAL.

   Experience shows that many SPRs do not contain enough information to duplicate or identify the problem. Concise, complete information helps DIGITAL give timely and accurate service to software problems.

   If you find an error in the documentation, send an electronic mail message to *manufacturing@digital.com*.

Installation on OpenVMS Systems

On OpenVMS systems, DASware is installed optionally as part of the DEComni API V3.1 installation procedure. You can either install DEComni API V3.1 as a separate product, or install it during the BASEstar Open V3.1 installation procedure. On OpenVMS systems, the only transport network provided by DASware V3.1 is RS232.

The information that follows is valid for both VAX and Alpha architectures, unless otherwise stated.

Preparing for DASware Installation

Before starting the installation procedure, complete the preparation requirements outlined in this section.

Software Requirements (BASEstar Open Environment)

If you are installing DASware in the BASEstar Open environment, check that the software products listed in Table 2-5 have been installed on your system.

Table 2-5: Pre-requisite Software for OpenVMS (BASEstar Open Environment)

Product Name	Comments
BASEstar Open Server for OpenVMS Version 3.1	Ensure that the BASEstar Open Server Device Connectivity has been installed. See the *BASEstar Open Server Inst. & Management Guide* for your platform for details.

Transport Network Software

You must also ensure that the appropriate transport network software has been installed and configured. Table 2-6 specifies the necessary transport network software for each supported transport network.

Table 2-6: Transport Network Software (OpenVMS Platform)

Transport Network	Transport Network Provider
RS232	LAT (provided with OpenVMS)

Disk Space Requirements

The table below lists the disk space required to install and run each DASware module.

Usage	Disk space (blocks)
Installation	2000
Permanent	2000

Backing Up Your System Disk

DIGITAL recommends that you back up your system disk before installing any software. For details of how to perform a system disk backup, refer to your OpenVMS documentation.

Files Installed on Your System

This section lists directories and files created by DASware.

SYS$COMMON:[SYSLIB]

DASTK_SHR.EXE

DASTK232_SHR.EXE

OMNI_INTEGRATOR16_DEFS_INCLUDE.H

OMNI_INTEGRATOR16_DEFS_INCLUDE.BAS

OMNI_INTEGRATOR16_DEFS_INCLUDE.PAS

OMNI_INTEGRATOR16_DEFS_INCLUDE.ADA

OMNI_INTEGRATOR16_DEFS_INCLUDE.FOR

OMNI_INTEGRATOR16_DEFS_INCLUDE.MAR

OMNI_INTEGRATOR16_DEFS_INCLUDE.PLI

OMNI_INTEGRATOR16_DEFS_INCLUDE.R32

ODS:[SCRIPTS]

DASTK_SCHEMA.COM

Getting Help and Reporting Problems

If an error occurs while you are using DASware, and you believe the error is the result of a problem associated with the product, take one of the following actions:

- If you have a basic or DECsupport Software Agreement, call your Customer Support Center (CSC). The CSC provides telephone support for high-level advisory and remedial assistance.
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you purchased the product within the last 90 days and you think the problem is casued by a software error, you can submit an SPR.

If you submit an SPR, please 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. 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, relevant data files and so forth.
4. Provide a list of the program.
5. If the program is longer than 50 lines, submit a copy of it on machine-readable media (floppy diskette or magnetic tape). If necessary, also submit a copy of the program library used to build the application.
6. Report only one program per SPR. This will facilitate a faster response.
7. Mail the SPR package to DIGITAL.

Experience shows that many SPRs do not contain enough information to duplicate or identify the problem. Concise, complete information helps DIGITAL give timely and accurate service to software problems.

If you find an error in the documentation, send an electronic mail message to *manufacturing@digital.com*.

Installation on Windows NT Systems

On the Windows NT platform, DASware features can only be accessed through BASEstar Open services (the DEComni API is not currently available).

Preparing for DASware Installation

Before starting the installation procedure, you should complete the preparation requirements outlined in this section.

License Registration

The installation procedure prompts you to supply a software key. You must enter this code to make DASware available on your system. Failure to do so implies restricted use of the product.

For information about how to obtain the software key, refer to your cover letter.

Software Requirements (BASEstar Open Environment)

If you are installing DASware in the BASEstar Open environment, check that the software products listed in Table 2-7 have been installed on your system.

Table 2-7: Pre-requisite Software for Windows NT (BASEstar Open Environment)

Product Name	Comments
BASEstar Open Server for Windows NT Version 3.1	Ensure that the BASEstar Open Server Device Connectivity has been installed. See the *BASEstar Open Server Inst & Management Guide* for your platform for details.

Transport Network Software

You must also ensure that the appropriate transport network software have been installed and configured. Table 2-8 specifies the necessary transport network software for each supported transport network.

Table 2-8: Transport Network Software (Windows NT Platform)

Transport Network	Transport Network Provider
RS232	Telnet or direct connection through serial communication port. (Telnet is implemented by the DAS module layered on TCP/IP).
TCP/IP	You can install TCP/IP services during installation of the Windows NT operating system.
OSI Transport	Not available.

Backing Up Your System Disk

DIGITAL recommends that you back up your system disk before installing any software. For details of how to perform a system disk backup, refer to your Windows NT documentation.

Installing DASware

To install DASware on a Windows NT system, you must be a member of the Administrators group.

To initiate the installation procedure, run the SETUP.EXE file from diskette 1 of the

distribution set. The DASware installation procedure does not copy any files from the distribution media but uses files which were copied on your system by the BASEstar Open Server Device Connectivity installation procedure. Installing DASware on your Windows NT system takes approximately 1 minute.

Deinstalling DASware

To deinstall DASware, double click on the “uninstaller DASware” icon in the BASEstar Open Program Manager.

Getting Help and Reporting Problems

If an error occurs while you are using DASware, and you believe the error is the result of a problem associated with the product, take one of the following actions:

- If you have a basic or DECsupport Software Agreement, call your Customer Support Center (CSC). The CSC provides telephone support for high-level advisory and remedial assistance.
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you purchased the product within the last 90 days and you think the problem is casued by a software error, you can submit an SPR.

If you submit an SPR, please 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. 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, relevant data files and so forth.
4. Provide a list of the program.
5. If the program is longer than 50 lines, submit a copy of it on machine-readable media (floppy diskette or magnetic tape). If necessary, also submit a copy of the program library used to build the application.
6. Report only one program per SPR. This will facilitate a faster response.
7. Mail the SPR package to DIGITAL.

   Experience shows that many SPRs do not contain enough information to duplicate or identify the problem. Concise, complete information helps DIGITAL give timely and accurate service to software problems.

   If you find an error in the documentation, send an electronic mail message to *manufacturing@digital.com*.

Installation on HP-UX Systems

BASEstar Open Server installs the DASware software for HP-UX during the installation procedure. The BASEstar Open Server for HP-UX Version 3.1 installation procedure also installs the BASEstar Open Device Connection Application Programming Interface (DEComni API).

This is the only software required to install DASware on your system. For further information, refer to the *BASEstar Open Server Inst. & Management Guide* for your platform.

Preparing for DASware Installation

Before starting the installation procedure, you should complete the preparation requirements outlined in this section.

Transport Network Software Requirements

You must ensure that the appropriate transport network software has been installed and configured. Table 2-9 specifies the necessary transport network software for each supported transport network.

Table 2-9: Transport Network Software (HP-UX Platform)

Transport Network	Transport Network Software
RS232	No pre-requisite software. TCP/IP services are required if telnet is used.
TCP/IP	TCP/IP services
OSI	OTS/9000 OSI transport for HP-UX

Disk Space Requirements (DASware-specific files)

The table below lists the disk space required to install and run each DASware module.

Usage (Kbytes)	/usr/opt	/var/opt
Installation/permanent	5600	1200

Backing Up Your System Disk

DIGITAL recommends that you back up your system disk before installing any software. For details of how to perform a system disk backup, refer to your HP-UX documentation.

Running the Installation Verification Procedure (IVP)

The DASware installation procedure does not execute an IVP as part of the installation. However, to run an IVP for the three protocols supported by DASware on HP-UX after completing the installation, execute the following commands:

```
# setenv SHLIB_PATH /usr/lib
# setenv NLSPATH /usr/lib/nls/msg/%N
# cd /var/opt/dastk_310/examples
# runivp_dastk 232
# runivp_dastk TCP
# runivp_dastk OSI
```

You can only run the OSI IVP if OTS/9000 has been configured. For information about how to configure the OTS/9000 OSI transport for running the IVP (loopback connection on the same machine) and for device connectivity, refer to Appendix B.

Files Installed on Your System

This section lists directories and files created by DASware.

/var/opt/dastk_310/examples:

dastk_ivp.c

dastk_ivp

runivp_dastk

/var/opt/dastk_310/include:

omni_integrator16_defs_include.h

/usr/opt/dastk_310/lib:

dastk_232init.o

dastk_tcpinit.o

libdastk.a

libdastkosi.a

dastk_osiinit.o

dastksts.cat

libdastk232.a

libdastktcp.a

/usr/opt/dastk_310/shlib:

libdastk.sl

libdastk232.sl

libdastkosi.sll

libdastktcp.sl

Getting Help and Reporting Problems

If an error occurs while you are using DASware, and you believe the error is the result of a problem associated with the product, take one of the following actions:

- If you have a basic or DECsupport Software Agreement, call your Customer Support Center (CSC). The CSC provides telephone support for high-level advisory and remedial assistance.
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you purchased the product within the last 90 days and you think the problem is casued by a software error, you can submit an SPR.

If you submit an SPR, please 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. 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, relevant data files and so forth.
4. Provide a list of the program.

5. If the program is longer than 50 lines, submit a copy of it on machine-readable media (floppy diskette or magnetic tape). If necessary, also submit a copy of the program library used to build the application.
6. Report only one program per SPR. This will facilitate a faster response.
7. Mail the SPR package to DIGITAL.

   Experience shows that many SPRs do not contain enough information to duplicate or identify the problem. Concise, complete information helps DIGITAL give timely and accurate service to software problems.

   If you find an error in the documentation, send an electronic mail message to *manufacturing@digital.com*.

PART I

Using BASEstar Open to Exchange Data Via Transport Networks

This part describes how you can exchange data through transport networks with partner applications using BASEstar Open services, and how you must configure BASEstar Open for use with DASware. It also provides a complete code example that you can run in the BASEstar Open environment.

The Command Language Interface (CLI) is used to create and operate the required BASEstar Open objects. However, you could use the Application Programming Interface (API), or the Graphical Configuration Utility (GCU) to perform the same tasks.

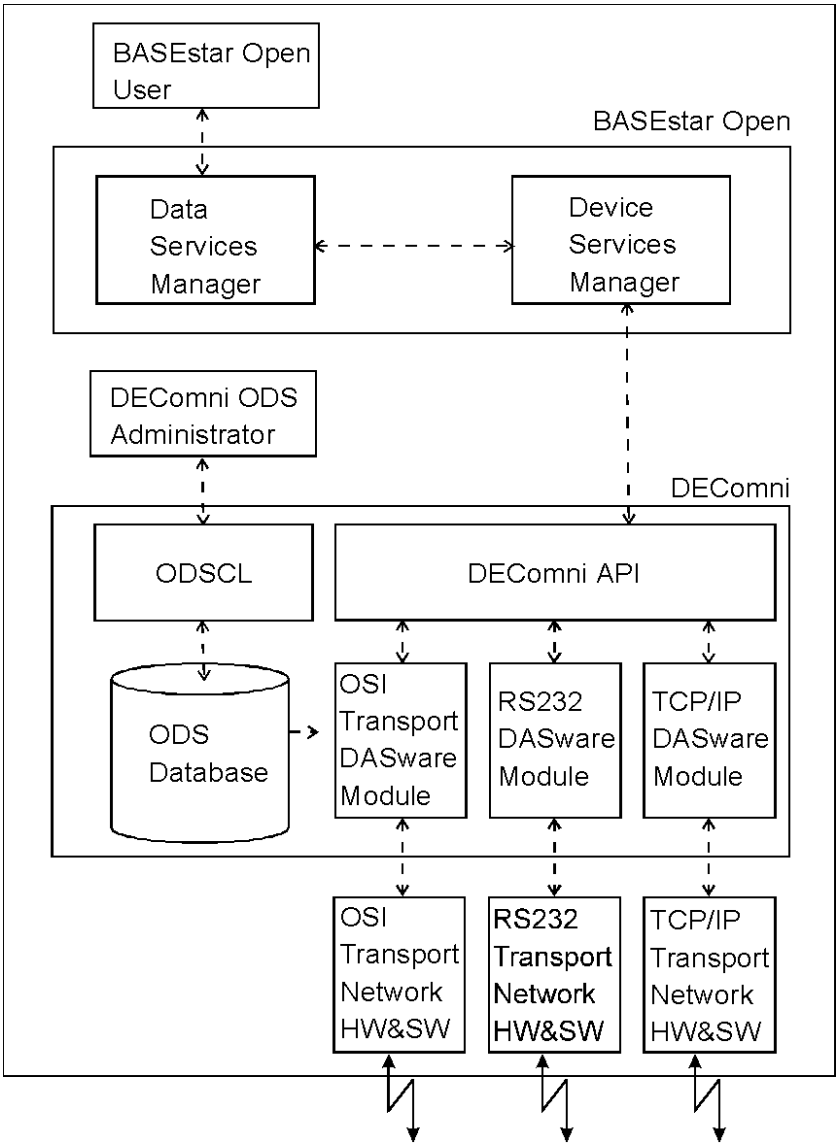
3
Overview

This chapter describes the BASEstar Open environment components that are involved when using BASEstar Open services to exchange data with transport network partner applications. It also provides an overview of the operations that can be performed by BASEstar Open users.

BASEstar Open Environment Components

Figure 3-1 shows the environment components involved when you exchange data through the transport networks supported by DASware.

Figure 3-1: BASEstar Open Environment Components



Data Services

BASEstar Open users use Device_Data_Points to send data to, and receive data from, a transport network.

When a user puts the value of a Device_Data_Point, BASEstar Open forwards a message to the transport network. Vice versa, BASEstar Open updates the value of a Device_Data_Point each time it receives a valid message from the transport network.

Each Device_Data_Point is linked to a Device Services variable which, in turn, is associated with the VMD configured for the selected transport network connection.

Device Services

BASEstar Open Device Services provide a unique set of operations to BASEstar Open users for device-independent access to any type of device or network.

This device-independent approach to device connectivity is possible because Device Services model each device as a MMS Virtual Manufacturing Device (VMD).

Each transport connection (for example, a TCP/IP connection) is mapped by a VMD. A variable associated with this VMD corresponds to a data item exchanged on the transport connection.

DEComni API

The DEComni API is the engine that provides BASEstar Open Device Services with MMS functions. DEComni also allows new DAS modules to be added to the BASEstar Open environment in a standardized way.

The Omni Directory Services (ODS) database is the DEComni component that allows you to define the information required for addressing the supported transport network.

Transport Network Software

For each transport network you wish to use, you must install the software product (and any hardware required) that support the transport network.

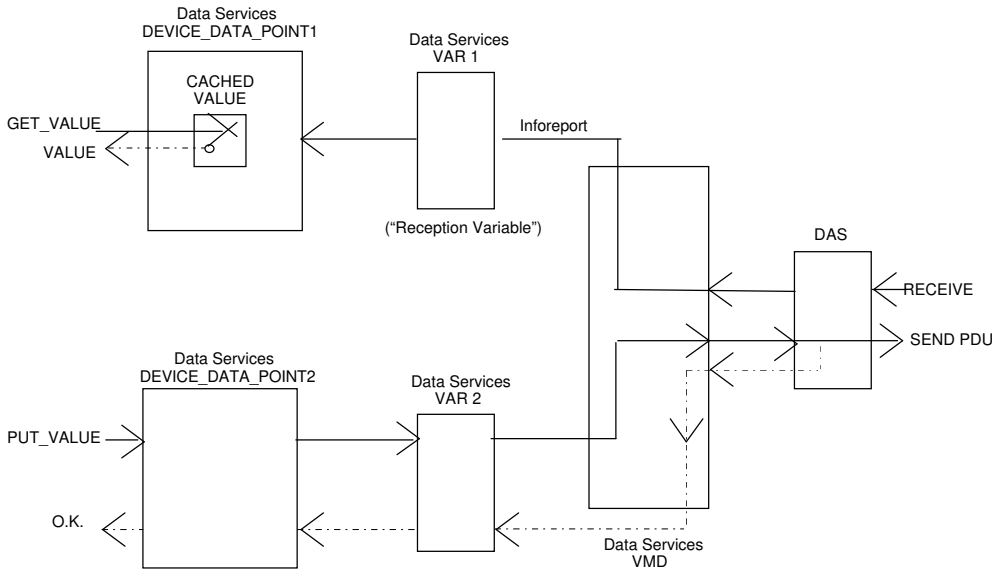
Before using a DASware module, ensure that you have already installed and configured the required transport network software on your system.

Exchanging Data Through a Transport Network

This section describes what happens when a BASEstar Open user uses a Device_Data_Point to send and receive data through a transport network.

Figure 3-2 shows the objects and commands a BASEstar Open user must use to receive data messages from (and send data messages to) a transport network partner application.

Figure 3-2: BASEstar Open Objects and Operations



Sending Data

To send a data messages to a transport network partner application, a BASEstar Open user must issue a PUT VALUE command on a Device_Data_Point.

As shown in Figure 3-2, the Device_Data_Point must be linked to a Device Services variable, which is, in turn, associated with a VMD that has been configured for accessing the desired partner application through the chosen transport network.

Before sending a message, DASware performs the following operations:

1. Obtains the data to be forwarded to the transport network from the user buffer, and encodes it according to the datatype associated with the linked Device Services variable, as explained in Section 4.3. (For example, a value of the INTEGER_16 datatype is encoded as 2 bytes.)
2. Depending on the transport network in use, adds specific information to the outgoing encoded data (see Table 3-1 for details).
3. Forwards the encoded data as a transport network specific Protocol Data Unit (PDU), for example, an RS 232 data stream or a TCP/IP message.

Note

Data is transmitted to the transport network according to the BIGENDIAN mode, that is, the MSB (Most Significant Byte) is sent first.

Table 3-1: Protocol-Specific Information Added to Outgoing Encoded Data (BASEstar Open)

Transport Network	Protocol-Specific Information
RS 232	UNIX DASware adds a trailer to the encoded data, consisting of a termination sequence that is specified by the value of the DTK232_TERM_STRING logical variable. For example, if the value of the DTK232_TERM_STRING logical variable is "13,10", the termination sequence consists of a Carriage Return character followed by a Line Feed. This sequence is transmitted to the line following the user data. Windows NT DASware adds a trailer to the encoded data, consisting of a termination sequence specified by the value assigned to the END_TX_SEQ keyword in the file %DASTK%\LIB\def.fsm. If no value is specified for the DTK232_TERM_STRING global variable, the CR,LF termination sequence is assumed by default.
TCP/IP	The TCP/IP DAS is internally registered as "WORD COUNTED". This means that, during data transmission, DASware adds a header consisting of a 16-bits counter that contains the length (in bytes) of the data being transmitted.
OSI Transport	No protocol-specific information is added.

Receiving Data

A BASEstar Open user must issue a GET VALUE command on a Device_Data_Point to obtain from BASEstar Open the value of a data message received from the Transport network partner application. The Device_Data_Point must have been configured for receive operations, and must be linked to the “reception variable” associated with the VMD that models the transport network connection toward the partner application.

BASEstar Open always uses the reception variable to return the data received from the transport network partner application.

The "reception variable" is the *first* Device Services Named_Variable that has been created for a given VMD. If no Named_Variables have been defined, the first Unnamed_Variable is used. If no variable is associated with the remote VMD, any data received from the partner application are lost.

The GET VALUE command returns to the user the cached value of the Device_Data_Point itself; that is, there is no interaction with the transport network. BASEstar Open updates the value of the Device_Data_Point regardless of the receive operations performed by the user. When receiving a message from the transport network, BASEstar Open generates an internal Inforeport that causes the unsolicited updating of the cached value of the Device_Data_Point that is linked to the Device Services "reception variable".

BASEstar Open interprets the data received from the transport network, (for example, an RS 232 data stream or a TCP/IP message), according to the datatype associated with the "reception variable", as follows:

1. Removes any protocol-specific information from the data (see Table 3-2 for details).
2. Decodes the received PDU according to the datatype associated with the reception variable (see Section 4.3), and stores the decoded data into the user buffer.

   Note that if an insufficient number of bytes is received, the value returned in the user buffer is unpredictable; if instead the number of the received bytes exceeds that required to meet the datatype, the exceeding bytes are lost.

Note

Data is received according to the BIGENDIAN mode, that is the MSB (Most Significant Byte) is received first.

Table 3-2: Protocol-Specific Information Removed from Incoming Data (BASEstar Open)

Transport Network	Protocol-Specific Information
RS 232	UNIX DASware considers that a data message has been received as soon as it encounters the termination sequence specified by the value of the DTK232_TERM_STRING logical variable. For example, if the value of the DTK232_TERM_STRING logical variable is "13,10", the DAS considers that a PDU has been received from the transport network when a Carriage Return character is encountered followed by a Line Feed. The termination sequence is discarded. Windows NT DASware considers that a data message has been received as soon as it encounters the termination sequence specified by the value of the END_TX_SEQ keyword in the file %DASTK%\LIB\def.fsm. If no value is specified for the DTK232_TERM_STRING global variable, the CR,LF termination sequence is assumed by default.
TCP/IP	The TCP/IP DAS is internally registered as "WORD COUNTED". This means that the first two bytes of the TCP/IP PDU contain a 2-byte counter. The TCP/IP DAS always discards the first two PDU bytes after having received the PDU and before decoding it.
OSI Transport	No protocol-specific information is removed.

Encoding and Decoding Transport Network Data

Table 3-3 shows the coding and decoding rules BASEstar Open uses when sending and receiving data through a transport network.

The number of bytes involved in a send or receive operation depends on the datatype associated with the Device Services variable that is linked to the Device_Data_Point in use, therefore:

- For a send operation, each table entry specifies the number of user data bytes that will be included in the transport network PDU.
- For a receive operation, each table entry specifies the number of bytes that DASware takes from the received PDU before preparing the data to be passed to the user.

Table 3-3: Coding and Decoding Data by Datatype (BASEstar Open)

Datatype	Exchanged as
BOOLEAN	1 byte
TIME	Not supported
VISIBLE_CHAR	1 byte
INTEGER_8	1 byte
INTEGER_16	2 bytes
INTEGER_32	4 bytes
UNSIGNED_8	1 byte
UNSIGNED_16	2 bytes
UNSIGNED_32	4 bytes
F_FLOAT	4 bytes (IEEE float format)
D_FLOAT	Not supported
BCD1, BCD2, BCD3, BCD4, BCD5, BCD6, BCD7, BCD8	Not supported
OCTET	1 byte
BIT	1 byte
OBJECT_STATUS	Not supported
OBJECT_NAME	Not supported
Array_Datatype	As many times as the number of elements it contains. A variable length array is treated as an array having a fixed length; the maximum length is assumed.
Structure_Datatype	Packing of the simple datatypes without padding.

4
Configuring BASEstar Open

This chapter describes how you must configure BASEstar Open in order to exchange data with a partner application through any of the transport networks supported by DASware.

To configure BASEstar Open, you must:

- Register the appropriate entries in the ODS database.
- Create the required Application Services Activity and Program objects.
- Create the required Device Services Protocol_Profile, VMD and Variable objects.
- Create the required Device_Data_Points objects.

Note

For an explanation of BASEstar Open object attributes whose values are not specified here, refer to the *BASEstar Open Command Language Interface* and to the *BASEstar Open Reference Guide*.

Registering ODS Entries

You must create an ODS entry containing transport network information for each transport network partner application with which you wish to exchange data. See Appendix A for details.

Creating Application Services Objects

You can use the DATADEV server or separate Device and Data servers. The DATADEV server makes both Data and Device Services servers available in a single process, thus simplifying configuration operations, optimizing performance, and allowing the same Device_Data_Point to be used for both send and receive operations.

Note

The DATADEV server is used throughout this manual, therefore you should modify the information and examples provided in this chapter if you want to use two separate Device and Data Services servers (for example, if your BASEstar Open version does not support DATADEV servers).

See also Chapter 5 for examples showing the commands used to create the required Program and Activity objects. For further details, refer to the *BASEstar Open Command Language Interface* and to the *BASEstar Open Reference Guide*.

Creating Device Services Objects

You must create the following objects, in the order listed below:

1. Protocol_Profiles
2. VMDs
3. Variables

Creating Protocol_Profiles

Device Services uses a Protocol_Profile to associate a VMD with the appropriate Transport network (that is, with the DASware module). You must create one Protocol_Profile for each transport network, and you can use it for all the VMDs that are associated with that transport network. Table 4-1 lists the Protocol_Profile attributes whose values are specific to the supported transport networks.

Table 4-1: Protocol_Profile Attributes (BASEstar Open)

Attribute Name	Description and Values
-APPLPROFID	The identifier of the DAS to be used. Possible values are: • 151 for RS 232 • 152 for TCP/IP • 153 for OSI Transport
-MAXPDUSIZE	The maximum length of the Protocol Data Unit (PDU) exchanged on the transport network, including the length of any headers and trailers added by the DAS.

Example 4–1 shows the Protocol_Profiles for all the supported transport networks.

Example 4–1: Creating Protocol_Profile Objects (BASEstar Open)

```
BSTR> CREATE PROTOCOL_PROFILE PP_TNRS232 -APPLPROFID 151\
      -MAXPDUSIZE 1024 -LOG

BSTR> CREATE PROTOCOL_PROFILE PP_TNTCPIP -APPLPROFID 152\
      -MAXPDUSIZE 1024 -LOG

BSTR> CREATE PROTOCOL_PROFILE PP_TNOSI -APPLPROFID 153\
      -MAXPDUSIZE 1024 -LOG
```

Creating VMDs

A VMD is the gate that BASEstar Open makes available to its users to access a transport network partner. In fact, a DASware VMD is mainly characterized by the DVM_ACCESS_POINT attribute that must specify an ODS and a transport network Protocol_Profile.

You need as many VMDs as many ODS entries you have planned to access a given transport network: this depends on your needs and on the protocol characteristics of the transport network in use.

Table 4-2: VMD Attributes (BASEstar Open)

Attribute Name	Description and Values
-DVM_ACCESS_POINT	A list of one or more descriptors, one for each access point clients use to access the VMD. An access point descriptor has the following format: *<application_simple_name>: <protocol_profile_name>* where: *application_simple_name* must specify an ODS entry that is consistent with the transport network in use and *protocol_profile_name* specifies a Protocol_Profile and thus a transport network.

Example 4–2 shows VMD objects that are valid for all the supported transport networks. The strings specified in the -MODEL and -VENDOR attributes are only used for documentation purposes.

Example 4–2: Creating VMD Objects (BASEstar Open)

```
BSTR> CREATE VMD VMD_TNRS232_001\
       -MODEL "DASware RS232" -VENDOR "OSFdirect"\
       -DVM_ACCESS_POINTS ("/cn=OSFdirect":PP_TNRS232)\
       -DESCRIPTION "For RS232 direct connection with XXX Device"\
       -LOG

BSTR> CREATE VMD VMD_TNRS232_002
       -MODEL "DASware RS232" -VENDOR "ANYtelnet"\
       -DVM_ACCESS_POINTS ("/cn=ANYtelnet":PP_TNRS232)\
       -DESCRIPTION "For RS232 telnet connection with AAA system"\
       -LOG

BSTR> CREATE VMD VMD_TNTCPIP_001
       -MODEL "DASware TCP/IP" -VENDOR "DASware"\
       -DVM_ACCESS_POINTS ("/cn=localhost.dastk":PP_TNTCPIP)\
       -DESCRIPTION "For TCP/IP direct connection"\
       -LOG
```

```
BSTR> CREATE VMD VMD_TNOSI_001
      -MODEL "DASware OSI" -VENDOR "DASware"\
      -DVM_ACCESS_POINTS ("/cn=localhost.dastk":PP_TNTCPIP)\
      -DESCRIPTION "For OSI connection with TSAP DTKOSICLT"\
      -LOG
```

Creating Variables

This section explains how you can create Device Services variables objects that will be linked to the Device_Data_Points used for exchanging data through a transport network. Note that the datatype associated with these variables drives BASEstar Open in encoding and decoding the data exchanged between BASEstar Open users and transport network partner applications. Because the datatype of the received data message often differs from the datatype of the sent data message, a separate description is provided for the configuration characteristics of a "reception variable" and for those of a variable used to send data messages.

Creating Variables for Receive Operations

To receive data from a transport network partner application you can use a Named_Variable or an Unnamed_Variable. The first Named_Variable you define at a given remote VMD is assumed to be the "reception variable", and will be used for read operations only. If no Named_Variables are defined, the first Unnamed_Variable is used.

Table 4-3 lists the attributes you must specify for a correct definition of a reception variable.

Table 4-3: Device Services Reception Variable Attributes (BASEstar Open)

Attribute Name	Description and Values
-DATATYPE *datatype_name*	Name of the datatype used by BASEstar Open to decode the received PDU. See Chapter 3 for a description of the valid datatypes.
-SEND *notification*	This attribute must be set to ALWAYS. This setting directs BASEstar Open to update the value of the linked Device_Data_Point in unsolicited mode.
-ACCESS *access_type*	Must be set to READ.
-ADDRTYPE *address_type*	Significant for Unnamed_Variables only; it must be set to UNCONSTRAINED.
-ADDRESS *address*	Significant for Unnamed_Variables only; it can be set to any value.

Creating Variables for Send Operations

You can define as many Named or Unnamed_Variables associated with the VMD as you need to send data to the partner application. Each of these variables can have a different datatype, so you can send different types of data to the transport network partner application.

Table 4-4 lists the attributes that you must specify for a correct definition of a transmission variable.

Table 4-4: Device Services Transmission Variable Attributes

Attribute Name	Description and Values
-DATATYPE *datatype_name*	Name of the datatype used by DASware to encode the sent PDU. See Appendix A for a description of the valid datatypes.
-ACCESS *access_type*	Must be set to WRITE.
-ADDRTYPE *address_type*	Significant for Unnamed_Variables only; it must be set to

	UNCONSTRAINED.
-ADDRESS *address*	Significant for Unnamed_Variables only; it can be set to any value.

Creating Data Services Objects

BASEstar Open users exchange data messages with transport network partner applications by means of Device_Data_Points., without taking into consideration communication and device characteristics.

Creating Device_Data_Points for Receive Operations

Table 4-5 lists the values you must set to the Device_Data_Point attributes in order to receive data through a transport network.

Table 4-5: Device_Data_Point Attributes for Receive Operations (BASEstar Open)

Attribute Name	Description and Values
-VMD_NAME *vmd_name*	Must be set with the name of a transport network VMD.
-VAR_NAME *var_name*	Must be set with the name of the "reception variable" that is associated with the *vmd_name* VMD.
-DATATYPE *datatype_name*	Must be set with the name of the same datatype associated with the *var_name* variable specified by the -VAR_NAME attribute.
-DEVICE_ACCESS *device_access*	Must be set to READ to indicate that the linked Device Services variable produces that value of the Device_Data_Point.
-UPDATE_POLICY *update_policy*	Must be set to UNSOLICITED to indicate that Data Services updates the cached value of the Device_Data_Point as soon as the value of the linked producer Device Services variable changes (that is, each time BASEstar Open receives data from the transport network).

Creating Device_Data_Points for Send Operations

Table 4-6 lists the values you must set to the Device_Data_Point attributes in order to send data to a transport network partner application.

Table 4-6: Device_Data_Point Attributes for Send Operations

Attribute Name	Description and Values
-VMD_NAME *vmd_name*	Must be set with the name of a transport network VMD.
-VAR_NAME *var_name*	Must be set with the name of a "Transmission Variable" associated with the *vmd_name* VMD.
-DATATYPE *datatype_name*	Must be set with the name of the same datatype associated with the *var_name* variable specified by the -VAR_NAME attribute.
-DEVICE_ACCESS *device_access*	Must be set to WRITE to indicate that Data Services will use the provided value during the send operation of the Device_Data_Point to update the value of the linked Device

	Services variable (and therefore to cause BASEstar Open to forward a data message to the transport network).

5
A Complete Example

The example contained in this chapter shows you a possible BASEstar configuration that you can use to perform send and receive operations from/to any of the available transport networks.

Example Overview

Given that BASEstar Open is highly configurable, this example shows only one of many possible modes for implementing send and receive operations.

Send data operations

It is assumed that send operations are executed in synchronous mode. According to configuration, each time a user issues a PUT VALUE command, BASEstar Open immediately forwards a data message through to the involved transport network. The command only terminates and returns control to the user when the operation has been completed.

Receive data operations

In this example, BASEstar Open updates the value of the Device_Data_Point used for receive operations as soon as it receives data form the transport network. The user must keep reading the value of the same Device_Data_Point if he wants to know when its value changes due to reception of a new data message. However, you could change the BASEstar Open configuration in order to receive an Event each time BASEstar Open receives an unsolicited data message from the device; this would allow BASEstar Open users to be notified in an immediate and asynchronous mode of any change occurring on the device.

Before Running the Example

In addition to the general requirements described in Chapter 3 and Chapter 4 of this manual, the complete example requires that:

- The transport network in use has been configured and is functioning on the system on which the partner application is running.
- The transport network partner application or device with which you are exchanging data has been activated, can understand the received data and perform the appropriate actions. In the example, the partner application must be able to receive 4 bytes or 40 bytes of data, and must always answer with 4 bytes.

In addition, you must:

- Replace *my_tcpnode* with the TCP/IP name of your the BASEstar Open Node.
- Replace *my_DAS_id* with the identifier of the DAS (that is, the identifier of the transport network) you wish to use.
- Replace *ODS_Entry* with the name of the ODS entry that complies with the communication characteristics of the transport network in use.

CLI Source

Example 5–1 shows the CLI commands used to create BASEstar Open objects and to activate the required servers.

Example 5–1: Creating BASEstar Open Objects

```
! ----------------------------------------------------------------------

! The example is set up to work with the VOLATILE database,

! but you can easily adapt it to work with the PERMANENT database.

! ----------------------------------------------------------------------

SET SCOPE VOLATILE

! ----------------------------------------------------------------------

! Create Common Services  objects.

! ----------------------------------------------------------------------

CREATE NODE NODE1 -PHYSICAL_NAME "my_tcpnode"

CREATE ARRAY_DATATYPE ARRAY_OF_INTEGERS_16\
       -NUM_OF_ELEMENTS 10 -DATATYPE INTEGER_32

CREATE PROTOCOL_PROFILE PP_TN\
        -APPLPROFID My_DAS_id -MAXPDUSIZE 1024 -LOG

CREATE VMD VMD_DASWARE
       -MODEL "DASware" -VENDOR "DIGITAL"\
       -DVM_ACCESS_POINTS ODS_Entry:PP_TN)\
       -DESCRIPTION "<VARIABLE>(VMD for data exhcange)" -LOG

CREATE DOMAIN /DOM_DATA

CREATE ACTOR /ACT_SERVERS

! ----------------------------------------------------------------------

! Create the Application Services objects needed to define both the

! Data Services and Device Services servers.

! ----------------------------------------------------------------------

CREATE PROGRAM /ACT_SERVERS/PRG_DATA\
       -PROGRAM_KIND DATA

CREATE ACTIVITY /ACT_SERVERS/ACY_DATA\
       -PROGRAM /ACT_SERVERS/PRG_DATA -NODES (NODE1)\
       -DOMAINS (/DOM_DATA)\
       -STARTUP_TIMEOUT 120 -RECOVERY_POLICY NECESSARY

CREATE PROGRAM /ACT_SERVERS/PRG_DEVICE\
       -PROGRAM_KIND DEVICE

CREATE ACTIVITY /ACT_SERVERS\ACY_DEVICE\
       -PROGRAM /ACT_SERVERS/PRG_DEVICE -NODES (NODE1)\
       -VMDS (/VMD_DASWARE)\
       -STARTUP_TIMEOUT 120 -RECOVERY_POLICY NECESSARY

! ----------------------------------------------------------------------

! Execute the ACTOR to activate the Data and Device Services servers.

! ----------------------------------------------------------------------

EXECUTE ACTOR /ACT_SERVERS

! ----------------------------------------------------------------------
```

```
! Create the Device Services objects.
! You can change the datatypes associated with the variables to meet
! your application dialogue requirements.
! ---------------------------------------------------------------------

CREATE NAMED_VAR VMD_DASWARE.VAR_RCV
       -DATATYPE INTEGER_32\
       -SEND ALWAYS -ACCESSMODE READ

CREATE NAMED_VAR VMD_DASWARE.VAR_SEND001\
       -DATATYPE INTEGER_32\
       -ACCESSMODE WRITE

CREATE NAMED_VAR VMD_DASWARE.VAR_SEND002\
       -DATATYPE ARRAY_OF_INTEGERS_32\
       -ACCESSMODE WRITE

! ---------------------------------------------------------------------

! Create the Data Services objects

! ---------------------------------------------------------------------

CREATE DEVICE_DATA_POINT DP_RECEIVE\
       -VMD_NAME VMD_DASWARE\
       -VAR_NAME VAR_RCV -VAR_CLASS NAMED\
       -DATATYPE INTEGER_32\
       -DEVICE_ACCESS READ -UPDATE_POLICY UNSOLICITED
       -DEFAULT_VALUE 0

CREATE DEVICE_DATA_POINT DP_SEND001\
       -VMD_NAME VMD_DASWARE\
       -VAR_NAME VAR_SEND001 -VAR_CLASS NAMED \
       -DATATYPE INTEGER_32\
       -DEVICE_ACCESS WRITE

CREATE DEVICE_DATA_POINT DP_SEND002
       -VMD_NAME VMD_DASWARE\
       -VAR_NAME VAR_SEND002 -VAR_CLASS NAMED \
       -DATATYPE INTEGER_32\
       -DEVICE_ACCESS WRITE
```

Exchanging Data Through the Transport Network

You can now receive and send data through the transport network. A possible operation sequence that uses the objects previously created is described in Example 5–2.

Example 5–2: Exchanging Data Through the Transport Network

```
BSTR> PUT VALUE DEVICE_DATA_POINT DP_SEND001

(Integer 32) : 345

       .........

       ..”Keep reading until you receive the answer!”......

       .........

BSTR> GET VALUE DEVICE_DATA_POINT DP_RECEIVE

12
```

PART II

Using the DEComni API to Exchange Data Via Transport Networks

6
Overview

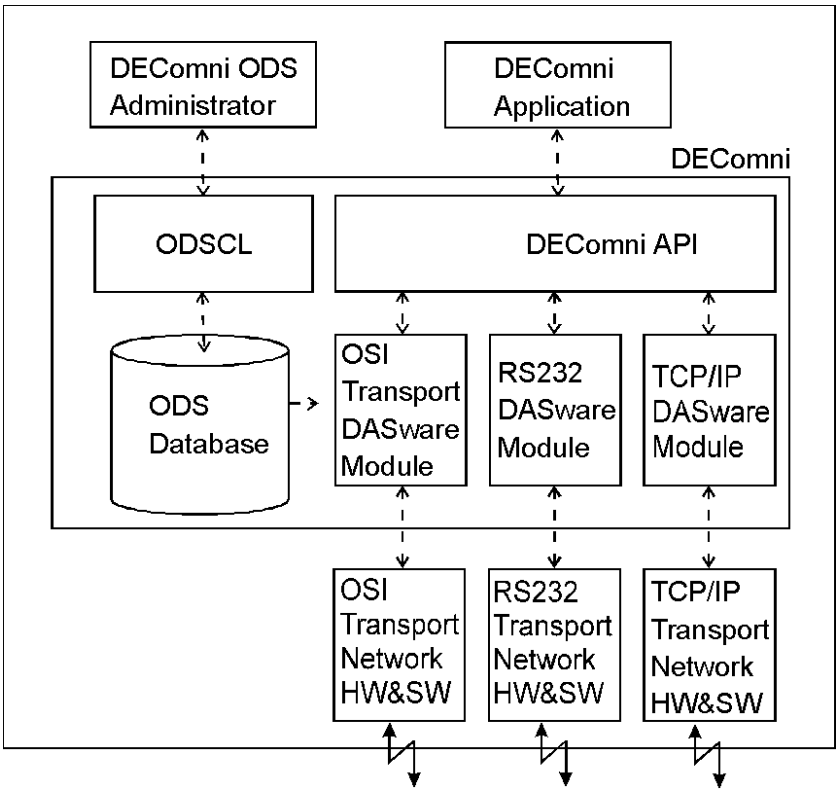
This chapter describes the DEComni environment components that are involved when DEComni API applications exchange data with transport network partner applications. Note that direct use of the DEComni API is limited to OpenVMS and DIGITAL UNIX platforms.

This chapter also provides an overview of the operations a DEComni API application can perform using the features provided by DASware. By calling the appropriate procedures, a DEComni API application can send data to, and receive data from a remote partner application.

DEComni API Environment Components

Figure 6-1 shows the components that are involved when DEComni API applications send and receive data through the supported transport networks.

Figure 6-1: DEComni API Environment Components



DEComni API

The DEComni API provides applications with a unique set of operations for device-independent access to any type of device or network. This device-independent approach to device connectivity is possible because each device is modeled as an MMS Virtual Manufacturing Device (VMD).

A VMD is associated with a connection that allows you to communicate with transport network partner applications. A variable object corresponds to data item exchanged with the partner application.

The DEComni API also allows new DAS modules to be added to the BASEstar Open environment in a standardized way.

The Omni Directory Services (ODS) database is the component of the DEComni API that allows you to define transport network addressing information.

Transport Network Software

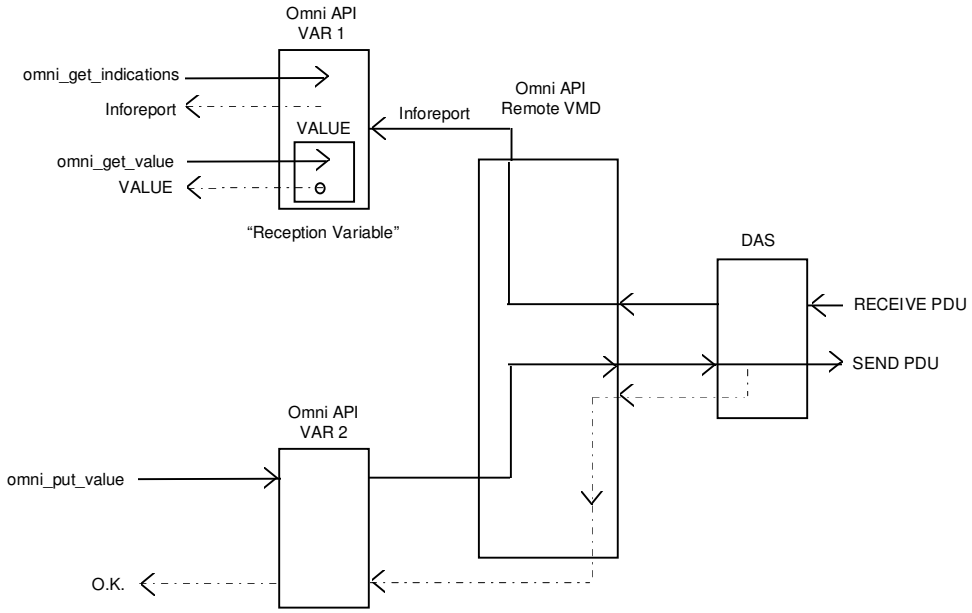
For each transport network you wish to use, you must install the software product (and any hardware required) that support the transport network. Before using a DASware module, ensure that you have already installed and configured the required transport network software on your system.

Exchanging Data Through the Transport Network

This section describes what happens when a DEComni API application sends and receives data through a transport network. It is assumed that the transport network connection has been established successfully.

Figure 6-2 shows the definitions and the API procedures a DEComni API application must use to exchange data with a transport network partner application.

Figure 6-2: DEComni API Definitions and Procedures



Writing Data Through the Transport Network

To send a data messages to a transport network partner application, an application must call the *omni_put_value*[_a] procedure, by specifying the handle of a named or an unnamed variable associated with a VMD that has been configured for the desired transport network. The *omni_put_value*[_a] call completes after the PDU has been delivered to the transport network provider.

On reception of the user data, the DAS:

1. Obtains the data to be sent through the transport network from the application buffer, and encodes it according to the datatype associated with the used variable (see Section 7.3).
2. Adds protocol-specific information (depending on the transport network in use) to the outgoing encoded data (see Table 6-1 for details).
3. Forwards the encoded data as a transport network-specific Protocol Data Unit (PDU); for example, an RS 232 data stream or a TCP/IP message.

Note

Data is transmitted to the transport network according to the BIGENDIAN mode, that is, the MSB (Most Significant Byte) is sent first.

Table 6-1: Protocol-Specific Information Added to Outgoing Encoded Data (DEComni API)

Transport Network	Protocol-Specific Information
RS 232	DASware adds a trailer to the encoded data, consisting of a termination sequence that is specified by the value of the DTK232_TERM_STRING logical variable. For example, if the value of the DTK232_TERM_STRING logical variable is "13,10", the termination sequence consists of a Carriage Return character followed by a Line Feed. This sequence is transmitted to the line following the user data. If no value is specified for the DTK232_TERM_STRING global variable, the CR,LF termination sequence is assumed by default.
TCP/IP	The TCP/IP DAS is internally registered as "WORD COUNTED". This means that, during data transmission, DASware adds a header consisting of a 16-bits counter that contains the length (in bytes) of the data being transmitted.
OSI Transport	No protocol-specific information is added.

Receiving Data from the Transport Network

To receive data from the transport network, an application must:

1. Call the *omni_get_indications*[_a] procedure, and specify a VMD that has been configured for the transport network.
2. When the *omni_get_indications*[_a] procedure returns an Inforeport indication, call the *omni_get_value*[_a] procedure on the same VMD, to obtain the data received from the transport network.

When receiving a message from the transport network, DASware generates an Information report that causes the completion of the *omni_get_indications*[_a] procedure call. The Inforeport indication returns the handle of the "reception variable" with which DASware has associated the received data.

A "reception variable" is the first named variable that has been created for a given VMD. This variable is always used by DASware to return the data received from the transport network.

If no Named_Variable has been defined, the first Unnamed_Variable is used. If there is not any variable associated with the VMD, any data DASware receives from the transport network are lost (and the *omni_get_indications*[_a] procedure returns an error).

Data received from the transport network in one message is automatically interpreted by the device access software according to the datatype associated with the "reception variable".

On reception of a PDU from the transport network, (for example, an RS 232 data stream or a TCP/IP message), the DAS performs the following operations:

1. Removes any protocol-specific information from the data (see Table 6-2 for details).
2. Decodes the received PDU according to the datatype associated with the reception variable, as explained in Section 7.3, and stores the decoded data into the user buffer.

Note that if an insufficient number of bytes is received, the value returned in the user buffer is unpredictable; if instead the number of the received bytes exceeds that required to meet the datatype, the exceeding bytes are lost.

Note

Data is received according to the BIGENDIAN mode, that is the MSB (Most Significant Byte) is received first.

Table 6-2: Protocol-Specific Information Removed from Incoming Data (DEComni API)

Transport Network	Protocol-Specific Information
RS 232	DASware considers that a data message has been received as soon as it encounters the termination sequence specified by the value of the DTK232_TERM_STRING logical variable. For example, if the value of the DTK232_TERM_STRING logical variable is "13,10", the DAS considers that a PDU has been received from the transport network when a Carriage Return character is encountered followed by a Line Feed. The termination sequence is discarded. If no value is specified for the DTK232_TERM_STRING global variable, the CR,LF termination sequence is assumed by default.
TCP/IP	The TCP/IP DAS is internally registered as "WORD COUNTED". This means that the first two bytes of the TCP/IP PDU contain a 2-byte counter. The TCP/IP DAS always discards the first two PDU bytes after having received the PDU and before decoding it.
OSI Transport	No protocol-specific information is removed.

Encoding and Decoding Transport Network Data

Table 6-3 shows the coding (and decoding) rules used by DEComni when sending and receiving data through a transport network.

The number of bytes involved in a send or receive operation depends on the MMS type associated with the variable in use.

Table 6-3: Coding/Decoding Data According to MMS Types (DEComni API)

Datatype	Exchanged as
omni_c_mmstype_boolean	1 byte
omni_c_mmstype_bit_str	(*n* + 7)/ 8 bytes where *n* is the number of bits in the string
omni_c_mmstype_integer	1, 2 or 4 bytes, depending on omni_c_symbol_size, that specifies the size of the variable (8, 16, or 32 bits)
omni_c_mmstype_unsigned	1, 2 or 4 bytes, depending on omni_c_symbol_size, that specifies the size of the variable (8, 16, or 32 bits)
omni_c_mmstype_floating_point	4 bytes (IEEE float format)
omni_c_mmstype_real	Not supported
omni_c_mmstype_octet_str	*n* bytes (as returned by sizeof())
omni_c_mmstype_visible_str	*n* bytes (as returned by strlen() if varying or as returned by sizeof() otherwise)
omni_c_mmstype_generalized_time	Not supported
omni_c_mmstype_binary_time	Not supported
omni_c_mmstype_bcd	Not supported

7
Configuring DEComni API

You must configure DEComni in order to allow DEComni API applications to exchange data with partner application through any of the transport networks supported by DASware. Configuring DEComni means creating the required definitions using the following API procedures:

- *omni_define*
- *omni_get_definition*
- *omni_modify_definition*

As an alternative, you can configure the DEComni API using the ODF facility (if this is available on your platform).

To configure the DEComni API for DASware, you must:

1. Register the appropriate entries in the ODS database. ODS entries define the parameters that are specific to the transport network currently in use.
2. Set the application profile for the DAS module in use.
3. Create the required DEComni VMD and Variable definitions.

Note

For an explanation of the parameters regarding the DEComni API definitions whose values are not specified here, refer to the *DEComni API and DEComni MMS User Guide* for your platform.

Registering ODS Entries

You must create an ODS entry containing transport network information for each transport network partner application with which you wish to exchange data. See Appendix A for details.

Setting the Application Profile

An application must invoke the *omni_set_application_profile* procedure to specify the appropriate application profile. This ensures that the definitions created after this procedure is called, are associated with the specified device access software. Table 7-1 lists the available application profiles.

Table 7-1: DEComni API Application Profiles for DAS Modules

DASware Module	Application Profile Constant
RS 232	omni_c_app_profile_dtk232
TCP/IP	omni_c_app_profile_dtktcp
OSI Transport	omni_c_app_profile_dtkosi

The following example shows how you invoke the *omni_set_application_profile* procedure for TCP/IP.

```
status=omni_set_application_profile (omni_c_app_profile_dtktcp)
```

Creating VMD Definitions

A DASware VMD models a transport network connection. A DASware VMD is mainly characterized by the omni_c_attr_appl_simple_name attribute that must specify an appropriate ODS entry for the network you wish to use.

Table 7-2 lists all the mandatory attributes that you must specify for correct definition of a DASware VMD.

Table 7-2: VMD Attributes (DEComni API)

Attribute Name	Description and Values
omni_c_attr_name	The name of the VMD definition.
omni_c_attr_appl_simple_name	The name of an ODS entry that contains the addressing information required to access the transport network partner application.
omni_c_vmd_max_segment	The maximum length of the Protocol Data Unit (PDU) exchanged on the transport network, including the length of any headers and trailers added by DASware.

Creating Variable Definitions

To exchange data with a transport network partner application, you can use both named or unnamed variables.
The first named variable you define for a given VMD is assumed to be the "reception variable" and it will be used for receive operations only. If no named variables are defined, the first unnamed variable is used.

You can define as many named or unnamed variables as you need that are associated with the VMD to send data to the partner application. Each of these variables can have a different datatype, which allows you to send different types of data to the transport network partner application.

Table 7-3: Variable Attributes (DEComni API)

Attribute Name	Description and Values
omni_c_attr_app_type_descr	Name of the application type used by the DAS to decode (encode) the received (sent) PDU. The associated MMS type must comply with those shown in Chapter 7.
omni_c_attr_address_type	Only significant for Unnamed_Variables; it must be set to unconstrained.
omni_c_attr_address_string	Only significant for Unnamed_Variables; it can be set to any value.

8
Programming

This chapter describes how you can use the DEComni API procedures to write a DEComni application that exchanges data with partner applications through the supported transport networks. Bear in mind the following considerations when using the API procedures to write DASware applications:

- General syntax and programming rules, described in the *DEComni API and DEComni MMS User Guide* for your platform, are also valid when the API procedures are used in the DASware context.
- When using the *DEComni API and DEComni MMS User Guide* for your platform, ignore the descriptions referring to services and functions that are not supported by the DASware modules.

DEComni API Procedures for Data Exchange

Table 8-1 lists the DEComni API procedures that applications can invoke to send data to, and receive data from, partner applications..

Table 8-1: DEComni API Procedures for Data Exchange

Procedure Name	Description
omni_abort\[_a]	Performs the forced closure of a transport network connection.
omni_conclude[_a]	Performs the forced closure of a transport network connection. (It behaves like the *omni_abort*[_a] procedure).
omni_connect[_a]	Opens a transport network connection.
omni_get_indications[_a]	Obtains connection-related and Inforeport indications (data received from the transport network).
omni_put_value[_a]	Sends data to the transport network. It is an unconfirmed operation, that is the procedure completes as soon as the data have been delivered to the transport network

Opening and Closing a Transport Network Connection

To open a transport network connection, a DEComni API application must call the *omni_connect*[_a] procedure, and specify a VMD that has been configured appropriately. The information associated with ODS specifies the addressing information DEComni uses to open the specific transport network connection.

To close a transport network connection, an application can use the *omni_conclude*[_a] or the *omni_abort*[_a] procedure. Both these procedures cause a forced closure of the transport network connection.

The application should use the *omni_get_indications*[_a] procedure to monitor any incoming omni_c_ind_abort indications that notify that the connection has been terminated in a forced way for any reason (for example, by the partner or by the transport network itself).

Sending Data Through the Transport Network

To send data through the transport network, an application must call the *omni_put_value*[_a] procedure, and specify the handle of a named or an unnamed variable associated with a VMD that has been configured for the desired transport network.

The *omni_put_value*[_a] returns control to the application as soon as DEComni has delivered the message to the transport network provider.

The operations performed by DASware and the output sent to the transport network have been described in Chapter 7.

Receiving Data from the Transport Network

To receive data from the transport network, an application cannot directly call the *omni_get_value*[_a] procedure, but it must first call the *omni_get_indications*[_a] procedure. This because DASware modules generate an Inforeport each time they receive a message from the transport network provider.

Thus, an application that wants to wait for data coming from the transport network, must call the *omni_get_indications*[_a] procedure, and specify a VMD that has been configured for that transport network.

When the *omni_get_indications*[_a] procedure returns control to the application with an omni_c_ind_info_report indication, the application must issue the *omni_get_value* procedure to get the value of the "reception variable". (Remember that the returned indication specifies a context that refers to the "reception variable".)

The operations performed by the DAS when data is received from the transport network have been described in Chapter 7.

PART III

Appendices

A
ODS Entry Attributes for DASware

This appendix describes the ODS attributes (and their associated values) that allow you to create the ODS entries required by each transport network supported by DASware. The information provided in this appendix is valid regardless of whether you are using DASware features via BASEstar Open services or via the DEComni API.

An ODS entry contains the addressing parameters needed for the selected transport network, and must be specified in any VMD definitions that are used to access that transport network. The number of ODS entries that you must create for a given transport network depends on your needs and on the protocol characteristics of the transport network itself.

For details of how to use the ODS CLI, refer to the DEComni documentation included in your BASEstar Open documentation kit.

Creating ODS Entries for the RS 232 Transport Network

An RS 232 connection can be based on direct connection, LAT or Telnet port.

Through Direct Connection or LAT

The table below provides a description of each ODS attribute, together with its class and accepted values.

Values of ODS Attributes for RS 232 Transport Network

Attribute	Description and Allowed Values
oc	Identifier of the ODS object class. Must be set to DTK232.
portname	Name of the communication port. Examples of valid values are tty00, TT00, LTA100 and COM1.
baudrate	Any valid baud rate supported by the Operating System and the hardware interface. Examples of valid values are 1200 and 9600.
charlen	Number of data bits. Allowed values are 5, 6, 7 and 8.
stopbit	Number of stop bits. Allowed values are 1 and 2.
prtycheck	Parity. Allowed values are NONE, ODD or EVEN.
flowctrl	Flow control. Allowed values are NONE, HW (standing for RTS/CTS) and SW (standing for XON/XOFF).
modem	Modem presence. Allowed values are 0 (modem not present, direct connection), and 1 (modem present, port handles modem signal and handshaking).
stationid	Station ID for multipoint connections.

	Allowed values are 0, 1-255.
lev7conn	Specifies if the conection is handled with messages at application level. Must be set to 0 because DASware does not exchange messages for connection at application level.
rsptimeout	Timeout, expressed in seconds, for the reception of the response to a confirmed service, after which the service is terminated with failure. Not significant DASware.

Example A–1 shows the configuration file commands that are used to create ODS registrations.

Example A–1: ODS Entries for RS 232 Direct Connection and LAT

UNIX

```
reg dir name "/cn=OSFdirect" attr
"/oc=DTK232/lev7conn=0/portname=tty00

/baudrate=9600/charlen=8/stopbit=1/prtycheck=NONE/flowctrl=NONE/modem
=0"

reg dir name "/cn=OSFlat" attr "/oc=DTK232/lev7conn=0/portname=tty08

/baudrate=9600/charlen=8/stopbit=1/prtycheck=NONE/flowctrl=NONE/modem
=0"
```

OpenVMS

```
reg dir name "/cn=VMSdirect" attr
"/oc=DTK232/lev7conn=0/portname=TT00

/baudrate=9600/charlen=8/stopbit=1/prtycheck=NONE/flowctrl=NONE/modem
=0"

reg dir name "/cn=VMSlat" attr "/oc=DTK232/lev7conn=0/portname=LTA100

/baudrate=9600/charlen=8/stopbit=1/prtycheck=NONE/flowctrl=NONE/modem
=0"
```

The corresponding LAT configuration is:

```
$ mc latcp

  > create port LTA100:  /application

  > set port LTA100:  /node=TERSERVER/port=PORT_1
```

Windows NT

```
reg dir name "/cn=WNTdirect" attr
"/oc=DTK232/lev7conn=0/portname=COM1

/baudrate=9600/charlen=8/stopbit=1/prtycheck=NONE/flowctrl=NONE/modem
=0"
```

Through Telnet

The table below provides a description of each ODS attribute, together with its class and accepted values.

ODS Attributes for the RS 232 Transport Network (Telnet)

Attribute	Description and Allowed Values
oc	Identifier of the ODS object class. Must be set to DTK232.
portname	Name of the server port, expressed in the form *@server.ip.address:port*
baudrate	Any valid baud rate supported by the Operating System and the hardware interface. Examples of valid values are 1200 and 9600.
charlen	Number of data bits. Allowed values are 5, 6, 7 and 8.
stopbit	Number of stop bits. Allowed values are 1 and 2.
prtycheck	Parity. Allowed values are NONE, ODD and EVEN.
flowctrl	Flow control. Allowed values are NONE, HW (standing for RTS/CTS) and SW (standing for XON/XOFF).
modem	Modem presence. Allowed values are 0 (modem not present, direct connection), and 1 (modem present, port handles modem signal and handshaking).
stationid	Station ID for multipoint connections. Allowed values are 0, 1-255.
lev7conn	Specifies if the connection is handled with messages at application level. Must be set to 0 because the DASware modules do not exchange messages for connection at application level.
rsptimeout	Timeout, expressed in seconds, for the reception of the response to a confirmed service, after which the service is terminated with failure. Not significant for DASware.

Example A–2 shows the configuration file commands that are used to create ODS registrations.

Example A–2: ODS Entries for RS 232 Telnet

```
reg dir name "/cn=ANYtelnet" attr

"/oc=DTK232/lev7conn=0/portname=@127.0.0.1:2001/baudrate=9600/charlen
=8/

stopbit=1/prtycheck=NONE/flowctrl=NONE/modem=0/stationid=1"
```

Creating ODS Entries for the TCP/IP Transport Network

The table below provides a description of each ODS attribute, together with its class and accepted values.

ODS Attributes for the TCP/IP Transport Network

Attribute	Description and Allowed Values
oc	ODS object class identifier. Must be set to DTKTCP.
netaddress	The IP address of the target system. On HP-UX platforms, the *netaddress* attribute must be set to the value of the 802.3 MACaddress of the device.
portnumber	The TCP port number of the serving application on the target system.
lev7conn	Specifies if the connection is handled with messages at application level. Must be set to 0 because DASware does not exchange messages for connection at application level.
rsptimeout	Timeout, expressed in seconds, for the reception of the response to a confirmed service, after which the service is terminated with failure. Not significant for DASware.

Example A–3 shows the configuration file commands that are used to create ODS registrations.

Example A–3: ODS Entries for TCP/IP

```
reg dir name "/cn=localhost.dastk" attr\
"/oc=DTKTCP/lev7conn=0/netaddress=127.0.0.1/portnumber=9999"
```

Creating ODS Entries for the OSI Transport Network

The table below provides a description of each ODS attribute, together with its class and accepted values.

ODS Attributes for the OSI Transport Network

Attribute	Description and Allowed Values
oc	ODS object class identifier. Must be set to DTKOSI.
netaddress	The OSI hexadecimal address of the target system. On VMS systems, the address must be prefixed by the name of a TEMPLATE. On HP-UX systems, it must contain the IEEE 802.5 MAC address.
tsap	The Transport Service Access Point (TSAP) on the target system.
lev7conn	Specifies if the connection is handled with messages at application level. Must be set to 0 because DASware does not exchange messages for connection at application level.
rsptimeout	Timeout, expressed in seconds, for the reception of the response to a confirmed service, after which the service is terminated with failure. Not significant for DASware.

Example A–4 shows the configuration file commands that are used to create ODS registrations.

Example A–4: ODS Entries for the OSI Transport Network

```
reg dir name "/cn=DASTK:osf.system" attr\
"/oc=DTKOSI/lev7conn=0/netaddress=49002E08002B08002B/tsap=DTKOSISRV"
reg dir name "/cn=DASTK:vms.system" attr\
"/oc=DTKOSI/lev7conn=0/netaddress=MYTEMPLATE%49002EAA0004AA000421/tsa
p=DTKOSICLT"
reg dir name "/cn=DASTK:HP-UX.system" attr\
"/oc=DTKOSI/lev7conn=0/netaddress=080006010002/tsap=DTOSIDEV"
```

B

Configuring DASware OSI Transport Services on an HP-UX System

A pre-requisite for using the DASware OSI transport services software is that the HP OTS/9000 software is installed and running. This software implements the OSI stack on HP-UX systems.

Configuring OTS/9000 Software

In order to use the DASware OSI transport services software, you must first configure the OSI transport components of the OTS/9000 package. The sections that follow explain how to:

- Obtain Ethernet card information (such as the MAC address)
- Configure the OSI transport for an 802.3 LAN, using the Null Internet option
- Configure the OSI transport for an 802.3 LAN, using the Full Internet option
- Configure the OSI Transport so that you can run the Installation Verification Procedure (IVP).

Note that Null Internet and Full Internet are not available simultaneously.

Obtaining Ethernet Card Information

Follow these steps:

1. Run the System Administrator Manager (SAM) tool from the `root` account:

   ```
   # sam
   ```

   To invoke this tool, select the LAN option from the *osiadmin* utility (see below).

2. Select Networking and Communication.
3. Select Network Interface Cards.
4. Make a note of the following parameters (you will need to enter them during other configuration operations):
   - Card Name of the IEEE802.3 Ethernet card type (subsequently referred to as the Device Interface Name)
   - Station Address (subsequently referred to as the MAC address)

     To display this information, double click on the Ethernet card item. The System Administrator Manager displays a popup window.

Configuring the OSI Transport for an 802.3 LAN (Null Internet Option)

Follow these steps:

1. Run the OTS configuration utility *osiadmin* from the `superuser` account:

   ```
   # /etc/osiadmin
   ```

2. Select OTS from the *osiadmin* main menu.
3. Select Configure OTS. *osiadmin* displays the Set Configuration Mode window.
4. Click on Done to confirm the default values. *osiadmin* displays the OTS Configuration menu.
5. Select Configure for the Subnetworks. *osiadmin* displays the Subnetwork Configuration menu.
6. Select Add... for the CLS over 802.3 item. *osiadmin* displays the Add CLNS Over 802.3 window.
7. Enter the following information:
   - Subnetwork name -- type the name of the subnetwork (e.g. **ieee**)
   - Network ID -- leave blank
   - Local network address (NSAP) -- type the MAC address of the Ethernet card (e.g. **0800096D79A8**)
   - Device interface name -- type the device interface name of the Ethernet card (e.g. **lan0**)

     The MAC address and device interface name correspond to the Station Address and Card Name parameters of the System Administrator Manager tool.
   - CLNP subset -- type **0**. *osiadmin* displays a warning that the Null Internet option is not NIST-compliant; ignore this warning.
8. Click on Perform Task to confirm your entries. The changes do not take effect until you shut down and reboot the system. It is recommended that you complete the remaining configuration steps before shutting down the system.
9. Select Add... from the Destination Systems item of the OTS Configuration menu.

   WARNING

   It is recommended that you perform this configuration operation for all systems/devices that you wish to connect using the Null Internet option.

   osiadmin displays the Add Destination System window.
10. Enter the following information:
    - Network address (NSAP) -- type the MAC address of the remote system (e.g. **080006010002**)
    - Physical address of the remote system (IEEE MAC or X.121) -- type the MAC address of the remote system above
    - Outgoing subnetwork name -- type the subnetwork name specified in point 7 (e.g. **ieee**)
    - End System, Intermediate System or both? -- type **0**

- Request reverse charge? (y/n) -- leave blank.

11. Click on Perform Task.
12. Shut down and reboot the system.

Configuring the OSI Transport for an 802.3 LAN (Full Internet Option)

Follow these steps:

1. Run the OTS configuration utility *osiadmin* from the `superuser` account:

   ```
   # /etc/osiadmin
   ```

2. Select OTS from the *osiadmin* main menu.
3. Select Configure OTS. *osiadmin* displays the Set Configuration Mode window.
4. Click on Done to confirm the default values. *osiadmin* displays the OTS Configuration menu.
5. Select Configure for the Subnetworks. *osiadmin* displays the Subnetwork Configuration menu.
6. Select Add... for the CLS over 802.3 item. *osiadmin* displays the Add CLNS Over 802.3 window.
7. Enter the following information:
   - Subnetwork name --type the subnetwork name (e.g. **internet**)
   - Network ID -- leave blank
   - Local network address (NSAP) -- type the NSAP of the system (e.g. **4900010800096D79A821**).

     You can select any of the NSAP address formats. However, to facilitate interoperability between the two vendors, it is recommended that you observe the same default NSAP format used by DIGITAL DECnet/OSI. The default DIGITAL DECnet/OSI NSAP format is:

     ```
     49:xxxx:MAC address:21
     ```

     where *x* is any combination of 4 hexadecimal digits.

     For more detailed information, refer to the OTS/9000 *Installing and Administering OSI Transport Services* manual.
   - Device interface name -- type the device interface name of the Ethernet card (e.g. **lan0**).
   - CLNP subset -- type **3**

Click on Perform Task to confirm your entries.

Configuring the OSI Transport for Running the IVP

To configure the OSI transport for the local system, follow the instructions for the Full Internet option. However, configure the destination system as follows:

- Local network address (NSAP) -- type the NSAP of the system (e.g. **4900010800096D79A821**)
- Physical address of the remote system (IEEE MAC or X.121) -- type the MAC address of the system (e.g. **0800096D79A8**)
- Outgoing subnetwork name -- type the same subnetwork name specified in the configuration of the local system (e.g. **internet**)
- End System, Intermediate System or both? -- type **0**.