TAPEMANAGER

Cartridge Library Installation Guide

for

Unisys MCP Systems

Release 10.070.0641

June 2023

Dynamic Solutions

INTERNATIONAL

Copyright

Correspondence regarding this document should be addressed to:
Dynamic Solutions International, Inc.
Product Development Group
8744 Lucent Blvd. Suite 106 Highlands Ranch 80129
(800)641-5215 or (303)754-2000
Technical Support Hot-Line (800)332-9020
E-Mail: support@dynamicsolutions.com

Contents

Chapter 1
Overview

LibraryManager from *Dynamic Solutions International* (*DSI*) provides a standard interface for MCP tape management systems and user programs to access and manage Virtual Tape Libraries (VTL) and automated cartridge libraries. LibraryManager works with most tape formats (like most LTO). Cartridge and virtual tape libraries from the smallest to the largest are supported.

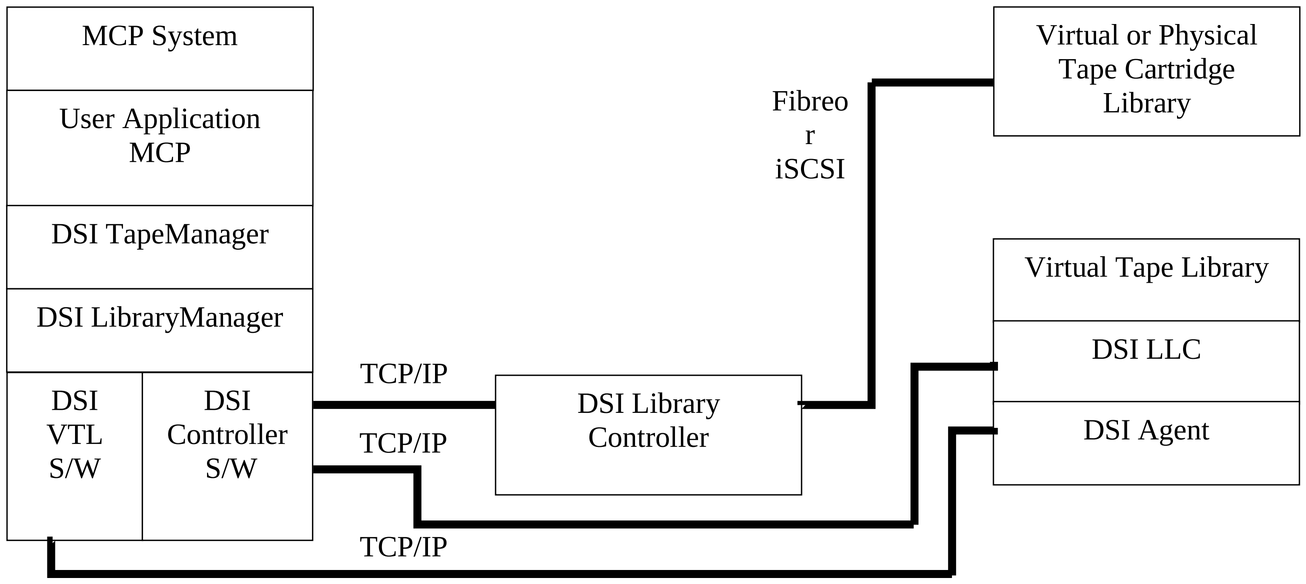
Features

- Supports many types of virtual tape and cartridge libraries including LTO-3, LTO-4, LTO-5, LTO-6, and LTO-7
- Supports all sizes of cartridge libraries from 10 to 64,000 cartridges
- Supports multiple automated libraries
- Supports multiple MCP system hosts accessing one library
- Integrated with DSI TapeManager and select third party tape management software
- Common interface (API) provided for all types of virtual tape and cartridge libraries
- Easy installation and setup
- No patches to MCP or other software

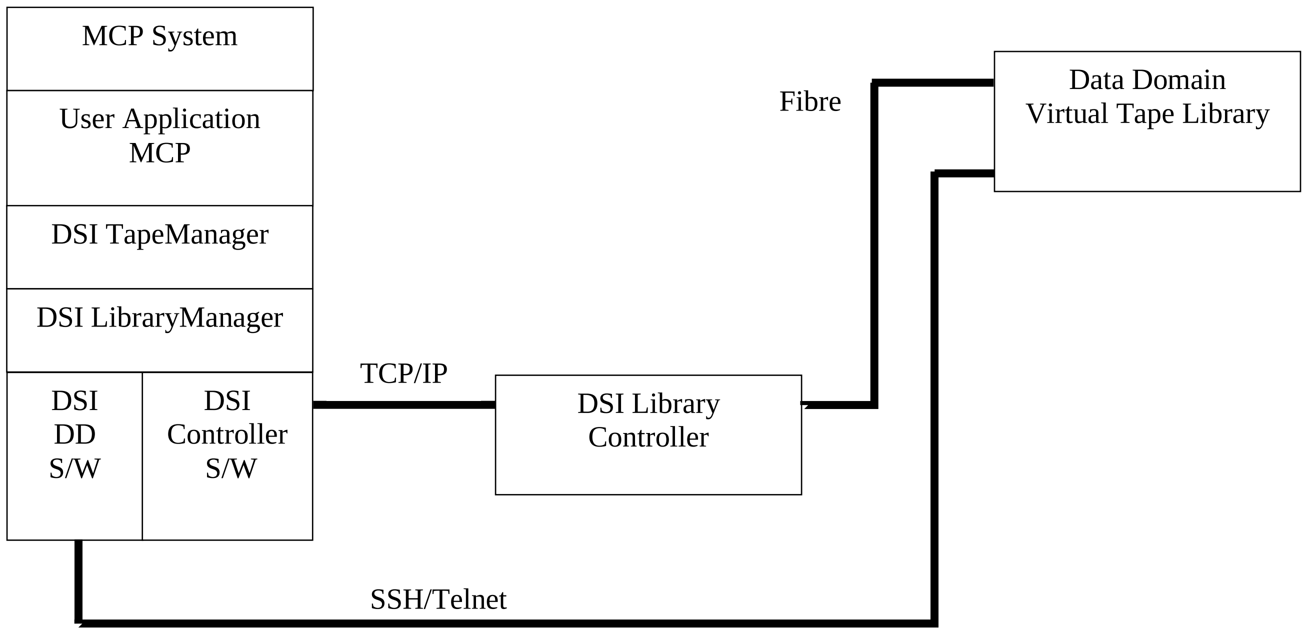
Requirements

- DSI Library Controller (also known as LLC)
- DSI VTL Agent (for DSI VTLs)
- A qualified virtual tape or physical cartridge library

Tape Library System Components



Tape Library System Components for Data Domain VTL



User Application

The user application is any software that requests the use of a data cartridge. These can be user written programs or software that is supplied with the operating system. These programs do not need any changes or special coding in order to use a cartridge library. Simply requesting a data cartridge by opening a file (KIND = TAPE) will allow the application to make use of a cartridge library.

Master Control Program (MCP)

One of the operating systems used on Unisys Corporation mainframes is known as the MCP. Some of the services that the MCP provides are I/O handling for programs. For programs requesting cartridge (TAPE) resources, the MCP also provides an optional TAPEMANAGER interface. If the user declares a tape management system to the MCP, the MCP will call the tape management system via this interface whenever tape activity (open, purge, etc.) is requested. When the tape management system is also linked to a cartridge library system, an appropriate cartridge can be automatically placed in a drive unit (virtual or physical) to handle the request.

Tape Management System

Tape Management systems track the usage and content of data cartridges. A tape management system can also track a cartridge’s location, prevent incorrect purging, and report cartridges available for use among other services.

The TapeManager from DSI provides these services as well as a tight integration with the LibraryManager software. The TapeManager provides a command interface to the LibraryManager so that operators and users can have complete control over the cartridge library from the operator’s console or terminal.

TapeManager may also be used with other tape management systems such as B&L Lib from B&L Associates. When used with another tape management system, TapeManager will act as the interface to the LibraryManager providing cartridge library support for these systems.

For more detailed information about TapeManager, please refer to the TapeManager Operations Guide.

Library Manager

The LibraryManager software module monitors and manages the access to one or more cartridge libraries. The LibraryManager presents a hardware/virtual independent interface to its callers. The calling tape management or other software need not be concerned with the size, media type, or the manufacturer of the cartridge library.

The LibraryManager maintains the status and inventory of all attached virtual tape or cartridge libraries. LibraryManager will periodically check the virtual tape or cartridge library for changes in status and cartridge inventory.

DSI Controller Support Library

The DSI Controller Support library is an MCP software library that contains the interface procedures for communicating with the DSI Library Controller (a.k.a. LLC). The procedures provide an I/O interface for the LibraryManager or other software that needs to communicate with the DSI Library Controller.

DSI Library Controller

The DSI Library Controller (also known as Linux Library Controller) is a component connected to the MCP host system via TCPIP. The Library Controller receives commands from the host and translates them to SCSI commands for the cartridge library. Responses from the cartridge library are received by the Library Controller translated and returned to the host. The Library Controller uses TCPIP (port 5600) to communicate to the MCP host.

A separate hardware-based Linux Library Controller is similar to the TCPIP based Library Controller in the VTL except as follows; 1) the LLC is in a separate hardware box, 2) the separate hardware box can only communicate to one virtual library in the VTL.

Cartridge Library

The cartridge library is the “hardware” component that contains the data cartridges and the drive units. This could be a physical tape library or a Virtual Tape Library (VTL) or even a VM (VMware, AWS, Azure) declared as a VTL. At a minimum, a cartridge library will be made up of cartridge storage slots, cartridge drive units, and a robotic mechanism for moving data cartridges between storage slots and drive units. Many cartridge libraries will also have a cartridge import/export mechanism (sometimes called a “mailbox” or “door” or “vault”) for placing data cartridges into and removing data cartridges from the cartridge library. Where available for physical cartridge libraries, they should be configured with a bar code scanner to allow for a faster inventory process.

Cartridge libraries will vary greatly in size and capacity. A library may contain anywhere from 10 to 10,000+ data cartridges. The library may have from 1 to 100+ drive units. Data cartridges may be tape (4mm, 8mm, AIT, DLT, LTO, etc.) or optical (Magneto-Optical, CD-ROM, etc.). Some libraries even support mixed media types.

Multi-host MCP Cartridge Library Configurations

The LibraryManager software can support multiple MCP hosts accessing one or more cartridge libraries. In these configurations one MCP host will be the “server” host controlling a cartridge library. The other hosts will be “client” hosts requesting cartridge library service from the server host. An MCP host may be both a client and a server. Each host must have access to one or more of the drive units within the cartridge library.

Cartridge library requests received on a client host are routed to the server host for processing. The server host responds to the client when the action or inquiry has been performed. The client/server process is handled invisibly to the callers of the LibraryManager.

Multi-O/S Cartridge Library Configurations

The LibraryManager software can also support accessing libraries that are shared by MCP and other operating systems (OS). Shared libraries are usually managed by partitioning the library or having intermediate software controlling the libraries.

Partitioned libraries are configured such that each host/OS can only see those parts of the library that is assigned to it. A partition will usually have one or more tape units, one or more storage slots, and possibly import/export slots. For LibraryManager, a partitioned library is treated as a single library that contains only the components in its partition.

Software managed libraries have software running on a server that manages requests from various hosts. Generally, all hosts can see all library components (drives, slots, doors) but the management software prevents actions from one host from interfering with actions from another host. This can be seen on the DSI VTL. The FalconStor software is running on the Linux OS. See Table 1 in Chapter 4 to determine the interfaces supported by a library.

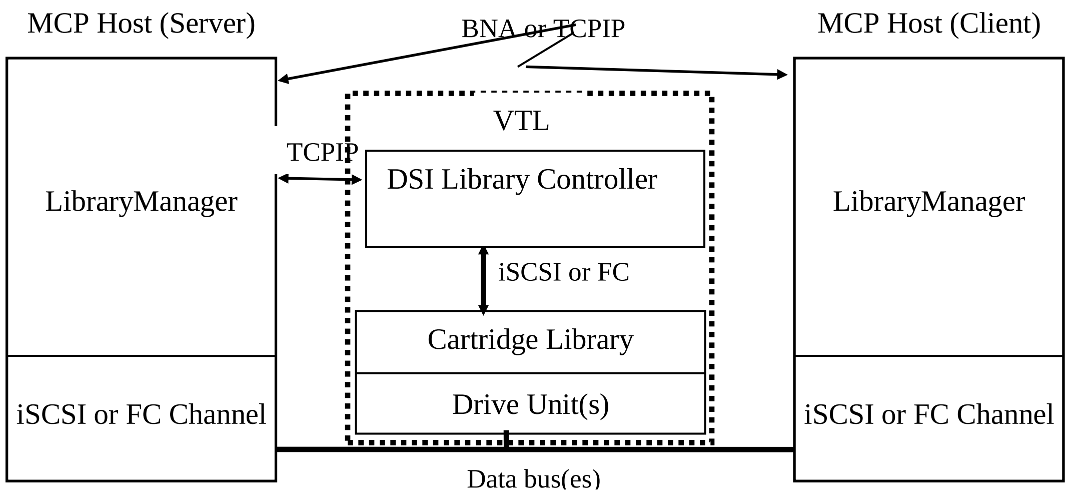


Figure 1 - Example of Multi-host MCP shared library

Chapter 2

The DSI Library Controller is software that allows an MCP system to communicate with a tape library (physical or virtual). The DSI Library Controller acts as a communications interface between the MCP system and a tape library. The DSI Library Controller receives commands from the MCP system, translates them to media changer commands, and passes them on to the tape library.

Linux Library Controller (LLC)

Setup of Linux Based Library Controllers

The Linux Library Controller (LLC) has two forms, the embedded form and the hardware form. The software is the same on the two forms. The embedded form comes pre-installed on current DSI VTL systems. The hardware form is a small server running Linux and the LLC software. The hardware form is used for non-DSI VTLs and physical tape libraries. The hardware form is sometimes referred to as an IPF due to its marketing style.

There is no setup for the embedded LLC at the VTL. It will use the same IP address as the VTL system. The LibraryManager configuration file on the MCP should be configured with a VTL statement in the library declaration. The name used in the VTL statement must match the name of the virtual library in the VTL exactly including case. If the VTL statement is missing, the LLC will connect to the first virtual library it finds. The VTL statement is required if an MCP host needs to connect to multiple virtual libraries in a single VTL.

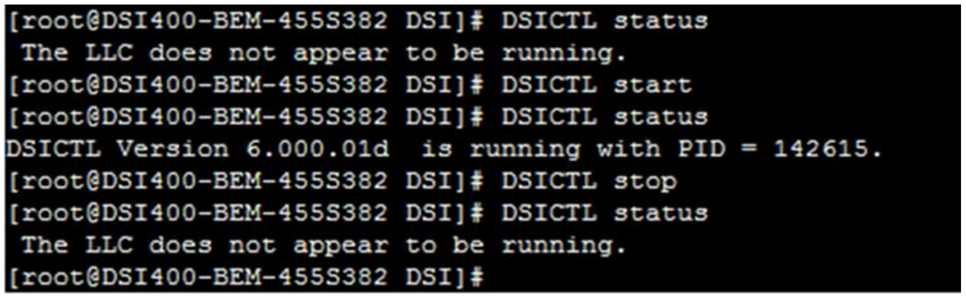
The hardware LLC only needs to have an IP address assigned to the hardware unit via Linux. The hardware LLC connects to the first media changer (library) that it finds on the Fibre Channel (FC) network. For this reason, it is highly recommended that FC zoning be used to connect the hardware LLC to the library.

There is no Configuration Options menu. The log and tracing features have set values. The log files are stored in the /usr/local/DSI/logs directory and the trace files are stored in the /usr/local/DSI/traces directory. The LLC limits logs and traces to 50 files each with each trace file limited to 10MB and each log file limited to 5MB.

Operation

There is very little operational control needed for the LLC. There are only a few commands available. The software starts automatically when the system is booted. The few commands available are.

- DSICTL start – starts the LLC software
- DSICTL stop – stops the LLC software
- DSICTL status – displays the current status of the LLC software



```
[root@DSI400-BEM-455S382 DSI]# DSICTL status
 The LLC does not appear to be running.
[root@DSI400-BEM-455S382 DSI]# DSICTL start
[root@DSI400-BEM-455S382 DSI]# DSICTL status
DSICTL Version 6.000.01d  is running with PID = 142615.
[root@DSI400-BEM-455S382 DSI]# DSICTL stop
[root@DSI400-BEM-455S382 DSI]# DSICTL status
 The LLC does not appear to be running.
[root@DSI400-BEM-455S382 DSI]#
```

Figure 3 – Linux Library Controller commands

Upgrading Linux Library Controller Firmware

Should the LLC need updating use the following steps. The process is the same for the embedded and hardware forms.

1. Download the current version of the Linux Library Controller found at http://www.dynamicsolutions.com/support/. On the software page under the TapeManager/LibraryManager section the most current version of the library controller software can be found. Downloading the Linux Library Controller will retrieve a file that looks like Install-DSICTL-0rr-x86_64.bsx file. (Where rr is the release level.)
2. Copy the downloaded file to the /tmp directory on the Linux system
3. Login to the Linux system as *root*
4. DSICTL stop (stops the LLC software)
5. cd /tmp (sets tmp as the current directory)
6. chmod 500 Install-DSICTL-0rr-x86_64.bsx (ensures the permissions are set correctly)
7. ./ Install-DSICTL-0rr-x86_64.bsx (runs the install program)
8. DSICTL start (starts the new LLC software)

Chapter 3
LibraryManager Software Installation

Installation starts by unwrapping the files from the TapeManager release container (.CON) file to an MCP system pack family. This package usually has the name *TapeManager_10.070.CON* and can be downloaded from our DSISupport (http://www.dynamicsolutions.com/support/) website. All the files must be unwrapped from the released .CON file to the same pack family. If the files are loaded under a usercode, all files must be under the same usercode. Be sure to remember to unwrap the files with RESTRICTED=FALSE.

Using the Install Program

LibraryManager is included as part of the TapeManager package. LibraryManager may be installed with or without the TapeManager software. An install program is provided as part of the package. Once the files have been copied to pack, the installation continues by running the SYSTEM/TAPEMANAGER/INSTALL program. This program will do the various SL and other system commands to setup the TapeManager and the LibraryManager systems. Once the installation program completes without error, a cartridge library configuration file must be created before the LibraryManager can be used.

Manual LibraryManager Installation

The LibraryManager will need to be installed manually if it was not included with a TapeManager package or if the installation program fails. The following files are supplied with the LibraryManager system:

SYSTEM/TAPELIBRARY/SUPPORT

This file is a library that is the core module of the LibraryManager system. It must be available at all times. The library should be active whenever there is the possibility of cartridge activity. This library must be SLed as TAPELIBRARYSUPP.

Example: SL TAPELIBRARYSUPP = SYSTEM/TAPELIBRARY/SUPPORT. The library code file is supplied with the MP +PU +CONTROL +LOCKED +IDENTITY command applied to it. (The library must be MP +PU at a minimum.)

SYSTEM/DSISUPPORT

This file is a support library used by the LibraryManager system and other products available from DSI. This library contains common procedures used by these software's. This library must be SLed as DSISUPPORT with the library attributes of ONEONLY and TRUSTED.

Example: SL DSISUPPORT = SYSTEM/DSISUPPORT: TRUSTED, ONEONLY. The library code file is supplied with the CONTROL, PU, LOCKED, and IDENTITY MP commands applied to it. (The library must be MP+ PU at a minimum.)

SYSTEM/DSICONTROLLER/SUPPORT

This file is a library that is used to communicate with the DSI Library Controller. It must be available at all times. The library should be active whenever access to the cartridge library is desired. This library must be SLed as DSICONTROLSUPPORT.

Example: SL DSICONTROLSUPPORT = SYSTEM/DSICONTROLLER/SUPPORT. The library code file is supplied with the MP +PU +CONTROL +LOCKED +IDENTITY command applied to it.

EXAMPLE/CONFIGURATION/=

These files are provided to give examples of LibraryManager configuration files. They are CANDE sequential text files.

Note: *All the above files are supplied as system files (non-usercoded) with a security of PUBLIC. The security must be changed if access to the system is to be restricted.*

The following file needs to be created as part of the tape library installation process.

SYSTEM/TAPELIBRARY/CONFIGURATION

This file describes the cartridge library configuration and its connections to the MCP system host.

The following files are only necessary if you are connected to that particular library type. For the DSI VTL it is VTLSUPPORT for an EMC Data Domain VTL it is DDSUPPORT.

SYSTEM/VTLSUPPORT

This file is a library that is used to communicate with DSI Virtual Tape Library (VTL) systems. It is only activated if there is a VTL statement for DSI VTLs in the tape library configuration file. This library must be SLed as VTLSUPPORT with the library attributes of TRUSTED and LINKCLASS = 1.

Example: SL VTLSUPPORT = SYSTEM/VTLSUPPORT: TRUSTED, LINKCLASS=1. The library code file is supplied with the MP +PU +LOCKED +IDENTITY command applied to it. (The library must be MP +PU at a minimum.)

SYSTEM/DDSUPPORT

This file is a library that is used to communicate with EMC Data Domain Virtual Tape Library (VTL) systems. It can only be copied to the system from that TapeManager package with a DSI-DDVTL software license. It is only activated if there is a VTL statement for Data Domain VTLs in the cartridge library configuration file. This library must be SLed as DDSUPPORT with the library attributes of TRUSTED and LINKCLASS = 1.

Example: SL DDSUPPORT = SYSTEM/DDSUPPORT: TRUSTED, LINKCLASS=1. The library code file is supplied with the MP +PU +LOCKED +IDENTITY command applied to it. (The library must be MP +PU at a minimum.)

Upgrading LibraryManager Software

The LibraryManager software will periodically need to be upgraded as enhancements and corrections become available. The upgrade process is similar to the initial install process.

1. Bring down the TapeManager or other cartridge management system that is calling the LibraryManager. This can usually be done by issuing the SEND TM QUIT or other command as required by the calling software. For the case of a DSI TapeManager you can bring down the LibraryManager from a DSI utility command line (not an MCP ODT) with the TM QUIT TL command. When there are no users (callers) of LibraryManager it will go to end of task. Do this when there is no cartridge activity to avoid missing updates to the TapeManager database.
2. Backup the previous LibraryManager program and configuration file (this would be SYSTEM/TAPELIBRARY/CONFIGURATION). Note – DSI TapeManager will automatically make a backup of this file under BACKUP/SYSTEM/TAPELIBRARY/CONFIGURATION when a TapeManager backup is performed.
3. Be sure your SYSTEM/KEYSFILE is up to date with DSI license keys. Check this by doing an IK SHOW "DSI" ODT command. If the proper keys are not installed, the needed software might not be unwrapped from the .CON file.
4. Unwrap the files from the LibraryManager (actually TapeManager) release container (.CON) file over the existing LibraryManager files (be sure when you unwrap the files you use the RESTRICTED=FALSE option).
5. Do not SL- any of these libraries in order to keep the properties already established for these libraries.
6. SL TAPELIBRARYSUPP to the new LibraryManager code file.
   Example: SL TAPELIBRARYSUPP = SYSTEM/TAPELIBRARY/SUPPORT.
7. SL DSISUPPORT to the new DSI Support library.
   Example: SL DSISUPPORT = SYSTEM/DSISUPPORT.
8. SL DSICONTROLSUPPORT to the new DSI Controller support library.
   Example: SL DSICONTROLSUPPORT = SYSTEM/DSICONTROLLER/SUPPORT.
9. SL VTLSUPPORT to the new DSI VTL support library.
   Example: SL VTLSUPPORT = SYSTEM/VTLSUPPORT.
10. SL DDSUPPORT to the new Data Domain VTL support library.
    Example: SL DDSUPPORT = SYSTEM/DDSUPPORT.

Note – Steps 6-10 are automatically performed when you run SYSTEM/TAPEMANAGER/INSTALL even for non-DSI TapeManager set ups.

11. Activate the TapeManager or other cartridge management system. This can usually be done by issuing a SEND TM START command or running the calling software as required. For the DSI TapeManager you can bring up LibraryManager by enabling a library (Ex. TM ENABLE LIB <Library name>).

Note: *If LibraryManager is received as part of the TapeManager product, the TapeManager Install program (SYSTEM/TAPEMANAGER/INSTALL) will automatically update the LibraryManager at the same time the TapeManager software is being updated.*

Chapter 4
LibraryManager Configuration File

A tape library configuration file must be created before the LibraryManager software will initialize. This file is used to describe the hardware configuration and communication interface of a cartridge library. More than one library may be declared in this file.

Creating the Configuration File

The tape library configuration file can be created and modified using the CANDE editor. The file is a standard CANDE sequential text (SEQ) file. It can be created with the CANDE MAKE command or by copying one of the EXAMPLE/CONFIGURATION/= files from the release package and modifying it to fit your configuration. The file name must be SYSTEM/TAPELIBRARY/CONFIGURATION. The file must be also on the same family and under the same usercode as the SYSTEM/TAPELIBRARY/SUPPORT program file.

Examples

CANDE: MAKE SYSTEM/TAPELIBRARY/CONFIGURATION SEQ
or
CANDE: GET EXAMPLE/CONFIGURATION/L700 AS
SYSTEM/TAPELIBRARY/CONFIGURATION

Configuration File Syntax

```
    ┌<────────────────────────────────────────────┐
────┴── <cartridge library declaration> ─────────┴───────┤
```

Cartridge Library Declaration

Syntax

```
<cartridge library declaration>

──┬── <begin library statement> ────────┬────────────────────────>
  └── <begin alternate statement> ──────┘
>─── <type statement> ───────────────────────────────────────────>
>─── <slots statement> ──────────────────────────────────────────>
>─── <doors statement> ──────────────────────────────────────────>
>─── <drives statement> ─────────────────────────────────────────>
>─── <connection statement> ─────────────────────────────────────>
>─┬─────────────────────────┬────────────────────────────────────>
  └── <VTL statement> ──────┘
>─┬── <end library statement> ──────────┬────────────────────────┤
  └── <end alternate statement> ────────┘
```

Explanation

The cartridge library declaration consists of a series of statements that describe the physical cartridge or virtual library and its connection to the MCP system host. Each statement must end with a semi-colon (;). Comments may also be placed in the file by using a percent sign (%) in the record. Any characters after the percent sign on that record are ignored by the LibraryManager software.

Begin Library Statement

Syntax

```
<begin library statement>

── BEGIN ── LIBRARY ── <library name> ──────────────────────────┤
```

Explanation

The BEGIN statement signifies the beginning of a cartridge library declaration and assigns a name to the library. The name can consist of up to 17 alphanumeric characters. The name may also include the dash (-) and underscore (_) characters but these must not be used as the first character of the name. TapeManager and LibraryManager use the name when reporting on the library. The name is also used in TapeManager commands when more than one cartridge library is configured to differentiate between libraries. If more than one library is declared, each library name must be unique.

A maximum of 9 libraries may be declared.

Begin Alternate Statement

Syntax

```
<begin alternate statement>

── BEGIN ── ALTERNATE ── <alternate name> ─────────────────────>

── FOR ─ <library name>──────────────────────────────────────────┤
```

Explanation

The BEGIN ALTERNATE statement signifies the beginning of an alternate cartridge library declaration and assigns a name to this alternate definition. The alternate name can consist of up to 17 alphanumeric characters. The alternate name may also include the dash (-) and underscore (_) characters but these must not be used as the first character of the name. The alternate name is used in the ENABLE LIBRARY USING <alternate name> command to specify which configuration should be used.

If an alternate library definition is declared, it must follow the definition of the <library name> otherwise a syntax error will be generated.

An example of creating an alternate library definition may be if the host is connected to the library via a remote connection, there may be a need to declare the host to be directly connected to the library in a failover condition.

A maximum of 20 libraries and alternates combined may be declared.

Type Statement

Syntax

```
<type statement>

─── TYPE ── = ── <library type> ──────────────────────────────┤
```

Explanation

The TYPE statement declares the actual hardware type of the cartridge library. The valid values for <library type> are shown in Table 1 and Table 2. It is important that the correct library type be declared, as various libraries require special handling by the LibraryManager.

Table 1
Physical Tape Library Attributes

Cartridge Library	Library Type	Number of Slots	Number of Doors	Number of Drives	Drive Type(s)	Port Type
Overland NEO	ALP60 or NEO	26-30	1	2	LTO2 or LTO3 or LTO4	SCSI
DSI 2000	DSI2000	26-60	0-2	2-4	LTO3 or LTO4 or SDLT600	SCSI (FC)
DSI 4000	DSI4000	56-120	0-4	2-8	LTO3 or LTO4 or SDLT600	SCSI (FC)
DSI8000	DSI8000	100-1000	15-30	2-24	LTO4	FC
DSI9000 VTL	DSI9000 or REO	1-180	0	1-16	LTO2	SCSI
IBM TS3500 (3584)	TS3500	58-6887	32-N	2-N	LTO5, 3592	FC
ATL P7000	P7000	399-679	12	4-16	DLT8000, SDLT320	SCSI
ATL P4000	P4000	100-322	12	4-10	DLT8000, SDLT320	SCSI
ATL P3000	P3000	170-326	12	4-16	DLT7000	SCSI
ATL P2000	P2000	100-198	12	2-10	DLT7000	SCSI
ATL P1000	P1000	16-30	1	2-4	DLT4000 or DLT7000	SCSI
ATL 4/52	DLT452	24-48	4	2-4	DLT4000 or DLT7000	SCSI
ATL 7/100	DLT7100	64-96	4	2-7	DLT4000 or DLT7000	SCSI
ATL 2640	DLT3264	264	1	2-3	DLT4000 or DLT7000	SCSI
ATL 6/176	DLT6176	176	1	3-6	DLT4000 or DLT7000	SCSI
ATL 9/88	DLT988	88	1	3-9	DLT4000 or DLT7000	SCSI
ATL L500	L500	14	0	2-3	DLT4000 or DLT7000	SCSI
STK L20	STKL20 or CLU20	18-20	0 or 1	2	DLT8000, LTO2, SDLT320	SCSI, ACSLS
STK L40	STKL40 or CLU40	20-41	2	2-4	DLT8000, LTO2, SDLT320	SCSI, ACSLS
STK L80	STKL80 or CLU80	20-81	5	2-8	DLT8000, LTO2, SDLT320	SCSI, ACSLS
STK L180	STKL180 or CLU180	84-174	10	2-6 (9840) 2-10 (DLT)	9840, DLT8000, SDLT320, LTO2	SCSI, ACSLS
STK SL500	STKSL500	24-575	0-5	2-18	LTO2, LTO3, LTO4, SDLT320, SDLT600	SCSI (FC), ACSLS
STK L700	STKL700 or CLU700	156-678	20-40	2-12 (9840) 2-20 (DLT)	9840, DLT8000, SDLT320,	SCSI, ACSLS

					LTO2,LTO3, LTO4, LTO5, LTO6, LTO7	
STK L5500	STKL5500	1500-5500	80	1-80	9840, LTO2	ACSLS
STK SL8500	STKSL8500	1448-7240	39-2496	64-2048	LTO2, LTO3, LTO4, SDLT600, 9840, 9940, T10000	ACSLS
STK 9360	STK9360 or CLU1000	504-949	20-30		9840 or 9490	ACSLS
STK 9310	STK9310 or CLU6000	2000-6000	21-80	1-80	9840 or 9490	ACSLS
STK 9710	STK9710	252-588	14	2-10	DLT4000 or DLT7000	SCSI, ACSLS
STK 9714	STK9714	40-100	1	2-6	DLT4000 or DLT7000	SCSI, ACSLS
STK 9730	STK9730	18-30	1	2-4	DLT4000 or DLT7000	SCSI, ACSLS
STK 9740	STK9740	326-494	14	2-10	DLT4000 or DLT7000	SCSI, ACSLS
BreeceHill Q2	BREECEHILLQ2	15	0	2	DLT2000 or DLT4000 or DLT7000	SCSI
BreeceHill Q4	BREECEHILLQ4	30	0	2-4	DLT2000 or DLT4000 or DLT7000	SCSI
BreeceHill Q7	BREECEHILLQ7	28	1	2	DLT2000 or DLT4000 or DLT7000	SCSI
BreeceHill Q47	BREECEHILLQ47	60	1	2-4	DLT2000 or DLT4000 or DLT7000	SCSI

Table 2
Virtual Tape Library Attributes

Library Type	Number of Slots	Number of Doors	Number of Drives	Drive Type(s)	Port Type
DSI640	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI620	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI540	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI520	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI420	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI400	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI370	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI350	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI300	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9253	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9303	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9983	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9152	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9252	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI
DSI9552	1-65535	1-65535	1-65535	LTO2, LTO3, LTO4, LTO5, LTO6, LTO7	FC or iSCSI

Slots Statement

Syntax

```
<slots statement>

── SLOTS ── = ── <integer> ─┬──────────────────────────┬──────────┤
                            └┬─ , ── <integer> ──────┬─┘
                             └───────────────────────┘
```

Explanation

The SLOTS statement declares the number of storage positions available in the cartridge library. If the number of slots declared does not match the number returned by the library, the smaller of the declared or reported value will be used. (See Tables 1 and 2 for the valid slot values.)

Some library types can be assembled in a modular fashion. Each module may contain a combination of resources (slots, drives, and doors). These modules are connected by a pass-thru mechanism. Declaring the modular configuration of the library allows the LibraryManager software to better utilize the library resources by first looking in the source module of the resource for the requested destination resource. Slots in a modular library are declared by using the <integer>, <integer>, … syntax. The modular syntax may only be used with library types known to allow modular configurations.

Note: *If SLOTS are declared as modules, then DOORS and DRIVES must also be declared with a matching number of modules. If a module does not have a particular resource, use zero (0) for that module.*

Note: *The maximum number of slots that can be declared in a single library is 64,000. The total slots declared for all libraries cannot exceed 65,535.*

Doors Statement

Syntax

```
<doors statement>

─── DOORS ── = ── <integer> ──┬──────────────────────────┬──────────>
                              │ ┌──────────────────────┐ │
                              └─┴── , ── <integer> ────┴─┘

        ┌──────────────── , ───────────┐
>── ( ──┴── NAME ── = ── <name> ───────┴── ) ──────────────────────┤
```

Explanation

The DOORS statement declares the number of input/output positions available in the cartridge library. The NAME clause must be repeated for each input/output position. The name can consist of up to 17 alphanumeric characters. The name may also include the dash (-) and underscore (_) characters but these must not be used as the first character of the name. Each name must be unique both within the library and between any other libraries declared. If the number of doors declared does not match that returned by the library, the smaller of the declared or reported value will be used. (See Tables 1 and 2 for the valid door values.)

Some library types can be assembled in a modular fashion. Each module may contain a combination of resources (slots, drives, and doors). These modules are connected by a pass-thru mechanism. Declaring the modular configuration of the library allows the LibraryManager software to better utilize the library resources by first looking in the source module of the resource for the requested destination resource. Doors in a modular library are declared by using the <integer>, <integer>, … syntax. The modular syntax may only be used with library types known to allow modular configurations.

Note: *If DOORS are declared as modules, then SLOTS and DRIVES must also be declared with a matching number of modules. If a module does not have a particular resource, use zero (0) for that module.*

Drives Statement

Syntax

```
<drives statement>

─── DRIVES ── = ── <integer> ──┬──────────────────────────┬──────>
                               │ ┌──────────────────────┐ │
                               └─┴─ , ── <integer> ─────┴─┘

                     ;
        ┌──────────────────────┐
>── ( ──┴── <unit statement> ──┴── ) ────────────────────────────┤

<unit statement>

─── UNIT ── = ── <unit number> ── , ─────────────────────────────>

>── TYPE ── = ── <unit type> ────────────────────────────────────>

>──┬─────────────────────────────────────────┬───────────────────>
   └── , ── SCSI-ID ── = ── <integer> ───────┘

>──┬─────────────────────────────────────────────────────────────┬┤
   └── , ── SHARED ──┬──────────────────────────────────────────┬┘
                     └─ WITH ── <host name> ── AS ── <integer> ─┘
```

Explanation

The DRIVES statement declares the number of tape or optical transport devices available in the cartridge library. The <unit statement> must be repeated for each transport. If the number of drives declared does not match the number returned by the library, the smaller of the declared or reported value will be used. The drives must be declared in the same order as they are referenced by the library hardware. That is library transport 1 must be the first unit described and so on. (See Tables 1 and 2 for valid drive values.)

Libraries that are shared by multiple hosts may have drives that are not visible to the host that this configuration file is processed on. These drives must still be declared in the configuration file so that the drive count and positions are correct.

The <unit number> is the MCP system unit number assigned to the tape or optical drive unit. This number is assigned in the PCD in the System Editor. Check with your support engineer for the correct unit number values. When declaring drives not visible to this host, use “dummy” unit numbers that are not used by this host. Valid unit numbers are 1 through 65,535.

The <unit type> describes the actual drive type being used. This identifier should describe the actual drive in-use not the type of drive that is being emulated if the drive is emulating a different unit type. It is important that the correct unit type be declared as various drives may require special handling by the LibraryManager. (See Tables 1 and 2 for valid unit type values.)

The SCSI-ID declares the actual SCSI address used by a unit on a SCSI bus. The SCSI-ID clause should only be declared for units that are on the same SCSI bus as the DSI Controller. The value must be between 0 and 255.

Note: *The SCSI-ID is not required and should not be declared. This is just a leftover implementation.*

Some library types can be assembled in a modular fashion. Each module may contain a combination of resources (slots, drives, and doors). These modules are connected by a pass-thru mechanism. Declaring the modular configuration of the library allows the LibraryManager software to better utilize the library resources by first looking in the source module of the resource for the requested destination resource. Drives in a modular library are declared by using the <integer>, <integer>, … syntax. The modular syntax may only be used with library types known to allow modular configurations.

Note: *If DRIVES are declared as modules, then SLOTS and DOORS must also be declared with a matching number of modules. If a module does not have a particular resource, use zero (0) for that module.*

Note: *The maximum number of drives that can be declared in a single library is 999. The total drives declared for all libraries cannot exceed 65,535.*

Connection Statement

Syntax

```
<connection statement>

─── CONNECTION ── = ──┬── <DSI control statement> ─┬──────────┤
                      ├── <remote statement> ──────┤
                      ├── <back end statement> ────┤
                      └── NONE ────────────────────┘
```

Explanation

The CONNECTION statement describes the method used by the LibraryManager to communicate with the automated cartridge library. If a library is directly connected to the host, a DSI Control is used. If the Multi-host option of LibraryManager is used, a remote library can be configured and accessed from the local host. The back-end connection is used for physical libraries that are attached to the back end (back-end library) of a VTL system.

The CONNECTION option of NONE is used with remote VTL systems where there is no robotics connection, but a VTL Agent is used to control functions at the remote VTL such as replicated tapes. If the NONE option is defined a VTL statement must be included in the library definition, or a syntax error will occur.

DSI Control Statement

Syntax

```
<dsi control statement>

─── DSICONTROL ── ( ─ ADDRESS ── = ── <TCPIP address> ───── , ─>

>── PORTTYPE ── = ───── SCSI ────────────────────────────── , ─>

>── PORT ── = ── <integer> ── ) ──────────────────────────────┤
```

Explanation

The DSICONTROL statement describes the physical connection of a cartridge library when a DSI Library Controller is used. This statement is used when the library is directly connected to this host. Use the REMOTE statement when the library is connected via another (server) host.

The DSI Library Controller connects to the MCP system via a TCPIPNATIVESERVICE port file. The ADDRESS clause defines the TCPIP network address that the DSI Library Controller will use.

The PORTTYPE and PORT are no longer used with the current Library Controller systems. However, they must be specified to keep compatibility with older configuration files. PORTTYPE must be SCSI and PORT can be any value from 0 to 255.

Remote Statement

Syntax

```
<remote statement>

─── REMOTE ── ( ── SERVICE ── = ─┬─ BNA ─┬─ , ────────────────>
                                 └─ TM ──┘

>── HOSTNAME ── = ── <name> ── ) ──────────────────────────────┤
```

Explanation

The REMOTE statement describes a cartridge library that is not physically connected to this host but is accessed via the LibraryManager Multi-host connection. To use this feature the local and remote hosts must be connected with a supported Multi-host service. Each host must also have a copy of the LibraryManager in operation.

The SERVICE value describes the inter-host communication service to be used. The SERVICE value of BNA causes LibraryManager to connect to the host with the tape library via BNA port file. The SERVICE value of TM causes LibraryManager to use TapeManager as its host-to-host communication service. The actual communication service used will depend on the communication service define with the CONFIGURE HOST TapeManager command. For example, use SERVICE = TM when TCPIP is required for Multi-host library usage.

The HOSTNAME is the BNA Hostname (as defined by the HN command) that is physically connected to the library. The HOSTNAME must be correctly defined even when using the TM service, which may not be using BNA.

Note: *The name of the library and its attributes must match that of the library definition on the local host that is physically connected to the library. This means that if the configuration file is changed on either the remote host or the host that is physically connected to the library that SYSTEM/TAPELIBRARY/SUPPORT needs to be brought down and back up on all hosts involved in order to reestablish unit tables.*

Note: *A local (server) host can support a maximum of 30 remote hosts.*

Backend Statement

Syntax

```
<backend statement>

── VTL ── = ── ( ── NAME ── = – <library name> ── ) ───────────────┤
```

Explanation

The VTL connection statement defines this library as a physical tape library connected to the back of a Virtual Tape Library (VTL) system. A backend library is used to import physical tapes to virtual tapes and export virtual tapes to physical tapes. The <library name> is the name of the virtual library that will use the backend library and is declared before this declaration.

The following restrictions apply to backend library declarations.

1) The VTL declaration referenced by <library name> must come before the backend library declaration.
2) The VTL library referenced by <library name> must have the VTL statement for connection to the VTL Agent.
3) The unit numbers used for the UNIT statements in the DRIVES declaration must not exist on the MCP host. (i.e., use dummy unit numbers)
4) Only one backend library may be declared for a virtual library.
5) Not all features and functions of directly connected physical libraries are available.

VTL Statement (for DSI Virtual Tape Libraries)

Syntax

```
<VTL statement>

─── VTL ── = ─── <VTL Console VTL name> ───────────────────────>

>── ( ── TYPE ── = ─ <VTL model> ── , ─────────────────────────>

>── ADDRESS ── = ── <TCPIP address> ── ) ──────────────────────┤
```

Explanation

The VTL statement defines this library as a logical virtual library declared in a Virtual Tape Library (VTL). This statement is used by both the embedded Linux Library Controller (LLC) and the DSI VTL Agent. LibraryManager will attempt to call the VTLSUPPORT library for each library with a VTL statement so that a connection can be made with the DSI VTL Agent to manage the logical library. LibraryManager will also pass this name to the LLC so it knows which logical virtual library to connect to.

The <VTL Console VTL name> is the name of the logical library as defined in the VTL. The name must match the logical library name as defined in the VTL exactly (including case) or the connection by the embedded LLC or the DSI VTL Agent will not be made.

The TYPE parameter is the model name of the VTL hardware (or DSI Restore). Currently the following models are supported: DSI640, DSI620, DSI540, DSI520, DSI400, and DSI420.

The ADDRESS parameter defines the TCPIP network address of the VTL hardware (or DSI Restore) where the logical library is defined.

VTL Statement (for Data Domain Virtual Tape Libraries)

Syntax

```
<VTL statement>

─── VTL ── = ─── <library name> ──────────────────────────────────>

>── ( ── TYPE ── = ─ <VTL model> ── , ────────────────────────────>

>── POOL ── = ─ <pool name> ── , ─────────────────────────────────>

>──┬─ CREDENTIALS ── = ─ <userid>/<password> ─────┬─ , ──────────>
   └─ SSHCREDENTIALS ── = ─ <userid>/<password> ──┘

>── ADDRESS ── = ── <TCPIP address> ── ) ─────────────────────────┤
```

Explanation

The VTL statement defines this library as a logical virtual library declared as a VTL in a Data Domain system. This statement is optional and should only be used when a connection to the Data Domain Command Line Interface (CLI) is desired to support ancillary functions. LibraryManager will attempt to call the DDSUPPORT library for each library with a Data Domain VTL statement so that a connection can be made with the Data Domain system to manage the logical library.

The <library name> is the name of the logical library as defined as a VTL in the Data Domain system. The name must match the VTL name exactly in the Data Domain system otherwise the control linkage will not be established.

The TYPE parameter is the model name of the Data Domain hardware. Currently the following models are supported: DD160, DD620, DD990, DD2200, DD2500, DD3300, DD4200, DD4500, DD6300, DD6800, DD6900, DD7200, DD9300, DD9400, DD9500. DD9800, DD9900, and DD9200.

The POOL parameter defines the storage area (also known as an Mtree) where the virtual tapes are stored. The <pool name> must match exactly the name of the pool assigned to the virtual library. All tapes in a Data Domain VTL must be in the same pool.

The CREDENTIALS or SSHCREDENTIALS parameter defines the user ID and password for accessing the Data Domain system. If the CREDENTIALS keyword is used, then a Telnet port is opened to the Data Domain system and login and commands are passed through that port. Since Telnet does not encrypt the data passed through the port this may not be a secure option for some sites. If the SSHCREDENTIALS keyword is used, then the Unisys MCP SSHCLIENT API is used to create a secure connection to the Data Domain system. Use of the SSHCLIENT API requires some preparation before it can be used (such as importing the Data Domain public key into the MCP SecurityCenter). Please see Unisys PLE 19058930 for additional information.

The ADDRESS parameter defines the TCPIP network address of the Data Domain hardware where the logical library is defined.

Data Domain implementation restrictions

1. *The VTL in the Data Domain system must be declared as model L180. No other emulations have been tested.*
2. *The virtual library name, the pool name, and the user ID can be a maximum of 64 characters. The allowable characters are A-Z (upper case), a-z (lower case), 0-9, hyphen (-), and underscore (_).*
3. *The password can be a maximum of 64 characters. The allowable characters are A-Z (upper case), a-z (lower case), 0-9, hyphen (-), underscore (_), equals (=), backslash (\), apostrophe (‘), tilde (~), exclamation point (!), ampersat (@), hash (#), dollar sign ($), caret (^), ampersand (&), asterisk (*), left parenthesis “(“, right parenthesis “)”, plus sign (+), pipe (|), open bracket ([), close bracket (]), open brace ({), close brace (}), semi colon (;), colon (:), period (.), forward slash (/), less than (<), greater than (>), and question mark (?). Note: the comma (,), blank ( ), and percent (%) characters are missing from this list because they are used for line continuation, formatting, and comment purposes.*
4. *The Data Domain vault name must match the pool name.*
5. *If multiple languages are supported by the Data Domain system only English is supported by DDSUPPORT.*
6. *All virtual tapes within a Data Domain virtual library must be of the same media type. The tapes do not have to be the same size.*
7. *All virtual tapes within a Data Domain virtual library must be in the same Data Domain pool. If replicating to another Data Domain system the pool name at the replication system must match the source pool name.*

End Statement

```
<end library statement>

─── END ── LIBRARY ── <name> ──────────────────────────────────┤
```

Explanation

The END LIBRARY statement signifies the end of a cartridge library declaration. The name must match the name given in the BEGIN LIBRARY statement.

End Alternate Statement

Syntax

```
<end alternate statement>

─── END ── ALTERNATE ── <alternate name> ──────────────────────┤
```

Explanation

The END ALTERNATE statement signifies the end of an alternate cartridge library declaration. The alternate name must match the name given in the BEGIN ALTERNATE statement.

Configuration Control Options

Configuration control options provide a means to control the handling of the SYSTEM/TAPELIBRARY/CONFIGURATION file. The options are similar to those used by the various compilers on the Unisys MCP systems.

Syntax

```
── $ ─┬────────────────────────────────────────────────────┬─┤
      │  ┌───────────────────────────────────────────────┐  │
      └──┴─────────────────────────────────────────────────┘
            ├─ SET ──┤    ├─ LIST ───────────────────────┤
            └─ RESET ┘    ├─ LISTDOLLAR ─────────────────┤
                          ├─ $ ──────────────────────────┤
                          ├─ LISTOMITTED ────────────────┤
                          ├─ OMIT ───────────────────────┤
                          ├─ HOSTNAME ─┬─────────────────┤
                          │            └─ = ── <hostname> ┤
                          └─ MCN ─┬──────────────────────┤
                                  └─ = ── <MCN> ─────────┘
```

Explanation

$

The dollar sign ($) is used to designate a configuration control card. The dollar sign must appear in column 1 or 2 of the record. A configuration control card may contain options that control the listing and parsing of the SYSTEM/TAPELIBRARY/CONFIGURATION file. All items on a configuration control card are considered options and not syntaxed as part of the library configuration. As with other configuration records, a comment marker (%) will stop the scanning of that record.

SET
RESET

SET and RESET control the Boolean value that is applied to all subsequent configuration control options in that record. Both SET and RESET may appear in the same record with the most recent value being applied to subsequent options. If neither, SET nor RESET has appeared in the record before an option is seen, SET is assumed.

Note: *Unlike the compiler implementation of these options, there is no "stack" of set/reset values. There is not a POP function to return an option to its previous value.*

LIST

When LIST is set, the configuration records will be written to a printer file (LINE). If an error is encountered in the configuration file, and LIST is set, the error information will also be written to the printer file. The default value for LIST is RESET.

LISTDOLLAR
$

When LISTDOLLAR ($) is set, and the LIST option is set, any configuration control records found will be written to a printer file (LINE). The default value for LISTDOLLAR is RESET.

LISTOMITTED

When LISTOMITTED is set, and the LIST option is set, configuration records that are skipped due to the OMIT or HOSTNAME option will be written to a printer file (LINE). If LISTOMITTED is reset, and LIST is set, skipped records are not written to the printer file. Records written to the printer file that are skipped will have the maker "<OMIT>" placed next to the record. The default value for LISTOMITTED is SET.

OMIT

The OMIT option can cause records in the configuration file to be skipped (ignored). When OMIT is set, all following configuration records, other than configuration control records, are skipped. If the LIST option is set, the setting of the LISTOMITTED option controls whether these records are written to the printer file. The default value for OMIT is RESET.

HOSTNAME = MYMCPHOST

The HOSTNAME option is used to set the value of OMIT based on a requested hostname and the host where the configuration file is being read. HOSTNAME = HOSTA will set the OMIT option if the processing host is not named HOSTA and reset OMIT if the processing host is named HOSTA. This option is equivalent to an Algol compiler card of SET OMIT = <my hostname> NEQ HOSTA. The = <hostname> part must not be used when RESET has been specified. RESET HOSTNAME is the same as RESET OMIT. The default value for HOSTNAME is RESET.

The purpose of the HOSTNAME option is to allow for a single configuration file to be used at multiple hosts where the configuration information is similar, such as local and remote hosts for the same library. In the following example, library DLTLIB is used by two systems named LOCALHOST and REMOTEHOST. LOCALHOST has a direct TCPIP connection to the library controller. REMOTEHOST accesses the library through its BNA connection to LOCALHOST. All other attributes are (and should be) the same. This file can now be copied to both hosts, but need be maintained only once.

```
$ SET LIST
BEGIN LIBRARY LTO5LIB;
    TYPE = STKSL500;
    SLOTS = 1000;
    DOORS = 1 (NAME = MBOXDOOR);
    DRIVES = 2 (UNIT = 81, TYPE = LTO5;
                UNIT = 82, TYPE = LTO5);
$ HOSTNAME = LOCALHOST
    CONNECTION = DSICONTROL (ADDRESS = 10.0.0.200,
                             PORTTYPE = SCSI,
                             PORT = 0);
$ HOSTNAME = REMOTEHOST
    CONNECTION = REMOTE (SERVICE = BNA,
                         HOSTNAME = LOCALHOST);
$ RESET HOSTNAME
END LIBRARY LTO5LIB;
```

MCN = 1234567890123456

The MCN option is used to set the value of OMIT based on a requested MCN and the host where the configuration file is being read. MCN = 1234567890123456 will set the OMIT option if the processing host does not have an MCN of 1234567890123456 and reset OMIT if the processing host has the matching MCN. This option is equivalent to an Algol compiler card of SET OMIT = <my MCN> NEQ MCN. The = <MCN> part must not be used when RESET has been specified. RESET MCN is the same as RESET OMIT. The default value for MCN is RESET. MCN stands for Manufacturing Control Number and is set by Unisys in the hardware at the time of manufacture.

The purpose of the MCN option is to allow for a single configuration file to be used at multiple hosts where the configuration information is similar, such as local and DR hosts for the same library. When using Unisys BCA (Business Continuity Assistant) the hostname may not change when doing a failover to the DR site. Therefore the $ HOSTNAME option will not work in a BCA failover. In the following example, the local and DR host both have VTL systems, but the network addresses differ. By using the MCN option the same file can be used at the DR host to activate the DR VTL. This file can now be copied to both hosts but needs to be edited only once.

```
$ SET LIST
BEGIN LIBRARY VTLLIB;
    TYPE = STKSL500;
    SLOTS = 1000;
    DOORS = 1 (NAME = MBOXDOOR);
    DRIVES = 2 (UNIT = 81, TYPE = LTO5;
                UNIT = 82, TYPE = LTO5);
$ MCN = 517253039
    CONNECTION = DSICONTROL (ADDRESS = 10.0.0.200,
                             PORTTYPE = SCSI,
                             PORT = 0);
$ MCN = 063D65E40000001F
    CONNECTION = DSICONTROL (ADDRESS = 10.0.0.100,
                                          PORTTYPE = SCSI,
                                          PORT = 0);
$ RESET MCN
END LIBRARY VTLLIB;
```

Note: To find your systems MCN you can use the **IK IPSHOW ALL** command or the **LS** command from the ODT. In order for LibraryManager to configure the library based on the MCN, SYSTEM/LICENSESUPPORT has to be running.

Testing a TapeLibrary Configuration File

The SYSTEM/TAPELIBRARY/CONFIGURATION file may be syntax checked before releasing into production. First, change the name of the file (or create it as) TEST/TAPELIBRARY/CONFIGURATION. This file must be under the same usercode and on the same pack family as the SYSTEM/TAPELIBRARY/SUPPORT program. To syntax check this file, run SYSTEM/TAPELIBRARY/SUPPORT; VALUE = 9. Setting the TASKVALUE to 9 causes the program to read the configuration file, check it for syntax, and exit rather than freeze as would be normal.

For configuration files that use $ HOSTNAME or $ MCN option cards, the configuration file can be checked for different hosts by assigning a host name or MCN to the TASKSTRING task attribute. For example, to check the syntax of the configuration file for the ABCHOST when running on a different host:

RUN $SYSTEM/TAPELIBRARY/SUPPORT; VALUE = 9; TASKSTRING = “HOSTNAME=ABCHOST”

To check the syntax of the configuration file for the host with an MCN of 1234567890123456 when running on a different host:

RUN $SYSTEM/TAPELIBRARY/SUPPORT; VALUE = 9; TASKSTRING = “MCN=1234567890123456”

LibraryManager Configuration File Examples

```
% Sample configuration of a CLU40 (STK L40) with 4 drives and 41
% slots
BEGIN LIBRARY CLU40;
     TYPE         = CLU40;
     SLOTS        = 41;
     DOORS        = 1 (NAME = D);
     DRIVES       = 4 (UNIT = 201, TYPE = DLT8000;
                       UNIT = 301, TYPE = DLT8000;
                       UNIT = 401, TYPE = DLT8000;
                       UNIT = 501, TYPE = DLT8000);
     CONNECTION   = DSICONTROL (ADDRESS = 10.0.0.102,
                                PORTTYPE = SCSI,
                                PORT = 0);
END LIBRARY CLU40;


% Sample configuration of two P3000's concatenated with pass-thru
% with 8 drives (4 in each cabinet), 24 doors (12 in each cabinet)
% and 652 slots (326 in each cabinet)
BEGIN LIBRARY DLTLIBRARY;
     TYPE         = P3000;
     SLOTS        = 326, 326;
     DOORS        = 12,12 (NAME = D1,  NAME = D2,  NAME = D3,
                           NAME = D4,  NAME = D5,  NAME = D6,
                           NAME = D7,  NAME = D8,  NAME = D9,
                           NAME = D10, NAME = D11, NAME = D12,
                           NAME = D13, NAME = D14, NAME = D15,
                           NAME = D16, NAME = D17, NAME = D18,
                           NAME = D19, NAME = D20, NAME = D21,
                           NAME = D22, NAME = D23, NAME = D24);
     DRIVES       = 4,4    UNIT = 201, TYPE = DLT7000;
                           UNIT = 202, TYPE = DLT7000;
                           UNIT = 203, TYPE = DLT7000;
                           UNIT = 204, TYPE = DLT7000;
                           UNIT = 221, TYPE = DLT7000;
                           UNIT = 222, TYPE = DLT7000;
                           UNIT = 223, TYPE = DLT7000;
                           UNIT = 224, TYPE = DLT7000);
     CONNECTION   = DSICONTROL (ADDRESS = 10.0.0.102,
                                PORTTYPE = SCSI,
                                PORT = 0);
END LIBRARY DLTLIBRARY;


% Example of an AIT-2 library connected to a remote host
BEGIN LIBRARY REMOTELIB;
    TYPE         = QUALSTAR;
    SLOTS        = 40;
    DOORS        = 1 (NAME = D);
    DRIVES       = 4 (UNIT = 91, TYPE = AIT2;
                      UNIT = 92, TYPE = AIT2;
                      UNIT = 81, TYPE = AIT2;
                      UNIT = 82, TYPE = AIT2);
    CONNECTION   = REMOTE (SERVICE = BNA,
                           HOSTNAME = OTHERA);
END LIBRARY REMOTELIB;
```

```
% Example of an alternate AIT-2 library configuration
% connected directly to the host for emergency fail over.  This
% configuration would be in the same configuration file as the
% above example.
BEGIN ALTERNATE LOCALLIB FOR REMOTELIB;
    TYPE         = QUALSTAR;
    SLOTS        = 40;
    DOORS        = 1 (NAME = D);
    DRIVES       = 4 (UNIT = 91, TYPE = AIT2;
                      UNIT = 92, TYPE = AIT2;
                      UNIT = 93, TYPE = AIT2;
                      UNIT = 94, TYPE = AIT2);
    CONNECTION   = DSICONTROL (ADDRESS = 10.0.0.102,
                               PORTTYPE = SCSI,
                               PORT = 5);
END ALTERNATE LOCALLIB;


% Example of a logical L700 in a VTL with the VTL Agent
BEGIN LIBRARY VL700;
    TYPE        =  STKL700;
    SLOTS       =  678;
    DOORS       =  1 (NAME = VDOOR);
    DRIVES      =  5 (UNIT = 3000, TYPE = LTO4;
                      UNIT = 3001, TYPE = LTO4;
                      UNIT = 3002, TYPE = LTO4;
                      UNIT = 3003, TYPE = LTO4;
                      UNIT = 3004, TYPE = LTO4);
    CONNECTION  = DSICONTROL (ADDRESS = 10.0.0.139,
                              PORTTYPE = SCSI,
                              PORT = 0);
    VTL         = VTLTST-1 (TYPE = DSI9252,
                            ADDRESS = 10.0.1.90);
END LIBRARY VL700;
```

```
% Example of a physical library connected to the back end of a VTL
BEGIN LIBRARY BACKEND;
    TYPE        =  DSI8000;
    SLOTS       =  29;
    DOORS       =  1 (NAME = BACKDOOR);
    DRIVES      =  2 (UNIT = 998, TYPE = LTO4;  % DUMMY UNIT NUMBER
                      UNIT = 999, TYPE = LTO4); % DUMMY UNIT NUMBER
    CONNECTION  = VTL (NAME = VL700);
END LIBRARY BACKEND;


% Example of a VTL in a Data Domain system
BEGIN LIBRARY DDLIB;
    TYPE        =  STKL180;
    SLOTS       =  500;
    DOORS       =  1 (NAME = DDCAP);
    DRIVES      =  5 (UNIT = 5000, TYPE = LTO3;
                      UNIT = 5001, TYPE = LTO3;
                      UNIT = 5002, TYPE = LTO3;
                      UNIT = 5003, TYPE = LTO3;
                      UNIT = 5004, TYPE = LTO3);
    CONNECTION  = DSICONTROL (ADDRESS = 10.0.1.143,
                              PORTTYPE = SCSI,
                              PORT = 0);
    VTL         = DDVTL (TYPE = DD990,
                         POOL = DDPOOL,
                         CREDENTIALS = ABC/XYZ,
                         ADDRESS = 10.0.1.144);
END LIBRARY DDLIB;
```

Chapter 5
VTLManager and VTL Agent

The DSI VTL Agent is an optional software component that is installed on the Virtual Tape Library (VTL) hardware. The VTL Agent is then accessed by client software on various hosts. A single instance of the VTL Agent may be accessed by multiple clients on multiple hosts. The VTLManager is the VTL Agent client for MCP hosts with TapeManager and LibraryManager software.

Using the Install Program

The VTLManager is included as part of a LibraryManager package, an install program is provided as part of the package. Once the files have been copied to pack, the installation continues by running the SYSTEM/TAPEMANAGER/INSTALL program. This program will do the various SL and other system commands to setup the TapeManager, LibraryManager, and the VTLManager systems. Once the installation program completes without error, a cartridge library configuration file that species a VTL connection must be created before the VTLManager can be used.

Manual VTLManager Installation

The VTLManager will need to be installed manually if it was not included with a TapeManager package or if the installation program fails. The following files are supplied with the VTLManager system:

SYSTEM/VTLSUPPORT

This file is a library that is the core module of the VTLManager system. It must be available to access the VTL Agent on a VTL. This library must be SLed as VTLSUPPORT. Example: SL VTLSUPPORT = SYSTEM/VTLSUPPORT. The library code file is supplied with the MP +PU +LOCKED +IDENTITY command applied to it. (The library must be MP +PU at a minimum.)

Note: *All the above files are supplied as system files (non-usercoded) with a security of PUBLIC. The security must be changed if access to the system is to be restricted.*

Upgrading VTLManager Software

The VTLManager software will periodically need to be upgraded as enhancements and corrections become available. The upgrade process is similar to the initial install process.

1. Bring down the LibraryManager which is calling the VTLManager. This is done by issuing the TM QUIT TL. When there are no users (callers) of VTLManager it will go to end of task.
2. Backup the previous VTLManager program.
3. Unwrap the files from the VTLManager (or TapeManager) release container (.CON) file over the existing VTLManager files.
4. SL VTLSUPPORT to the new VTLManager code file. Example: SL VTLSUPPORT = SYSTEM/VTLSUPPORT.
5. Activate the VTLManager and LibraryManager by issuing a TM ENABLE LIB command.

Note: *If VTLManager is received as part of the TapeManager/LibraryManager product, the TapeManager Install program will automatically update the VTLManager at the same time the TapeManager and LibraryManager software is being updated.*

VTL Agent Installation

The VTL Agent is installed on a DSI Virtual Tape Library system. The VTL-Agent software is distributed in a self-extracting compressed file. This file is **Install-VTL-Agent-2.03.016-x86_64.bsx**. Root privileges are required to install this software. Either log on to the DSI VTL system as "root" or issue an "su" if you are logged on as a different user. Copy the **Install-VTL-Agent-2.03.016-x86_64.bsx** file to a directory of your choice (i.e., **/usr/local** or **/tmp**).

Note*:* *On most systems the /tmp directory will be emptied when the system is rebooted. Choose a different directory if you want the install file to be available after a system reboot.*

Enter the command **ls –l Install-VTL-Agent-2.03.016-x86_64.bsx** to verify that the install file is local to your current directory. The results should be similar to:

```
   -rw-r--r-- 1 root root   70007 Apr 23 15:47 Install-VTL-Agent-2.03.016-x86_64.bsx
```

The file permissions are the first few characters of the result message. If the permissions do NOT start with "**-rwx**" enter the command:
chmod 744 Install-VTL-Agent-2.03.016-x86_64.bsx to give execute permission to the owner, which should be root. Re-enter the **ls –l Install-VTL-Agent-2.03.016-x86_64.bsx** command. The results you should now see will be:

```
   -rwxr--r-- 1 root root   70007 Apr 23 15:47 Install-VTL-Agent-2.03.016-x86_64.bsx
```

The install file will now be executable by the root user.

Run the install program by entering the name of the install file:
./Install-VTL-Agent-2.03.016-x86_64.bsx and enter. The script will uncompress the files and create the directory **/usr/local/agent**. Subdirectories of bin, scripts, logs, and conf are created at installation and execution time. The installation script links the script that starts and stops the program to the appropriate /etc/rc.d locations so that the program is stopped and started during a system shutdown and system start.

To start the VTL Agent manually enter **/usr/local/agent/scripts/Agent start.** To stop the VTL Agent manually enter **/usr/local/agent/scripts/Agent stop.** To check the running status of the VTL Agent enter **ps –e | grep VTL-Agent**. The response will report the process id (PID) in output similar to :

```
24875 pts/3    00:00:00 VTL-Agent
```

If the VTL Agent is not running you will be returned to the prompt. You can also ask for the Agent status by entering **Agent status**.

The VTL Agent software is installed in the **/usr/local/agent** directory. The executable is named **VTL-Agent** in the **/usr/local/agent/bin** directory. The scripts used with the VTL Agent are located in the **/usr/local/agent/scripts** directory. The VTL-Agent creates log files that are kept in the **/usr/local/agent/logs** directory. The VTL Agent writes to the current log file (**log**). When the current log file reaches a configurable size or age, the log is closed and renamed from **log** to **log1**. It will keep a configurable number of log files from 10 to 50 named **log** through **log49**. These configuration parameters are controlled by the client and are not available directly on the system that is running the VTL-Agent. When the VTL-Agent is stopped a **/usr/local/agent/conf/Agent.conf** directory and file are created. The Agent.conf file contains state information that is read when the VTL-Agent is restarted.

Chapter 6
DDManager

The DSI DDManager is an optional component that is used to access ancillary functions of a VTL configured within an EMC Data Domain system. It allows LibraryManager and the controlling media manager access to functions not available via the robotics interface.

Using the Install Program

The DDManager is included as part of the LibraryManager package. An install program is provided as part of the package. Once the files have been copied to pack, the installation continues by running the SYSTEM/TAPEMANAGER/INSTALL program. This program will do the various SL and other system commands to setup the TapeManager, LibraryManager, and the DDManager systems. Once the installation program completes without error, a cartridge library configuration file that specifies a Data Domain system connection must be created before the DDManager can be used.

Manual DDManager Installation

The DDManager will need to be installed manually if it was not included with a TapeManager package or if the installation program fails. The following files are supplied with the DDManager system:

SYSTEM/DDSUPPORT

This file is a library that is the core module of the DDManager system. It must be available to access the Command Line Interface of a Data Domain system. This library must be SLed as DDSUPPORT.

Example: SL DDSUPPORT = SYSTEM/DDSUPPORT. The library code file is supplied with the MP +PU +LOCKED +IDENTITY command applied to it. (The library must be MP +PU at a minimum.)

Note: *All the above files are supplied as system files (non-usercoded) with a security of PUBLIC. The security must be changed if access to the system is to be restricted.*

Upgrading DDManager Software

The DDManager software will periodically need to be upgraded as enhancements and corrections become available. The upgrade process is similar to the initial install process.

1. Bring down the LibraryManager which is calling the DDManager. This is done by issuing the TM QUIT TL. When there are no users (callers) of DDManager it will go to end of task.
2. Backup the previous DDManager program files.
3. Be sure you have the proper license keys for DDManager (DSI-DDVTL) checking **IK SHOW "DSI"**
4. Unwrap the files from the TapeManager release container (.CON) file over the existing DDManager files.
5. SL DDSUPPORT to the new DDManager code file. Example: SL DDSUPPORT = SYSTEM/DDSUPPORT.
6. Activate the DDManager and LibraryManager by issuing a TM ENABLE LIB command.

Note: *If DDManager is received as part of the LibraryManager product, the TapeManager Install program will automatically update the DDManager at the same time the TapeManager and/or LibraryManager software is being updated.*

DDSupport Help

The DDSupport Help available to users is found through the library interface. This Help is accessed by finding the mix number of SYSTEM/DDSUPPORT and then issuing an AX HELP. To find the mix number of SYSTEM/DDSUPPORT, from the ODT enter the command:
AA NAME=DDSUPP=
You should get a response back from the ODT like this:

```
---Mix-Pri--CPU Time------------ 1 ACTIVE ENTRY (ALL) =DDSUPP=-----------
6265 50 :00 Lib (TM)SYSTEM/DDSUPPORT ON DSIPACK
```

From this mix number (6265) you will see all the help commands that are supported:
6265AX HELP and then look at the system messages (**MSG**):

```
* 6265 09:35 DDVTL:<mix #>AX HELP [command name]
* 6265 09:35 DDVTL:<mix #>AX CONFIGURE <option>
* 6265 09:35 DDVTL:<mix #>AX MONITOR <bar code>
* 6265 09:35 DDVTL:<mix #>AX DEBUG [debug option]
* 6265 09:35 DDVTL:DDSUPPORT AX commands
```

To get a bit more help on the command in question you can use the HELP <command> in order to get a simple explanation of that command.

DEBUG

Looking at the DEBUG command you see the following help (in reverse order):

```
* 6265 09:45 DDVTL:   <option> = REPLICATION
* 6265 09:45 DDVTL:DEBUG <option> ON/OFF - sets/resets debug option
* 6265 09:45 DDVTL:DEBUG - displays current debug option settings
* 6265 09:45 DDVTL:Help for DEBUG command
```

To turn REPLICATION ON and cause extra messages in the sumlog for replication events (like Pre-comp Bytes Sent and Pre-comp Bytes Remaining) you would enter:
6265AX DEBUG REPLICATION ON

MONITOR

Looking at the MONITOR command in the HELP response you see (in reverse order):

```
* 6265 10:02 DDVTL:MONITOR <bar code> - checks replication for tape
* 6265 10:02 DDVTL:Help for MONITOR command
```

To monitor a specific bar code in the replication process you can enter that bar code here:
6265AX MONITOR DD0001

Note: Be sure the Data Domain library in Online

CONFIGURE

A feature allows for changing the timing used to check the Data Domain virtual tape library vault(s). The feature is accessed by doing an **AX CONFIGURE VAULT [seconds]** to the Control stack of the SYSTEM/DDSUPPORT library. The seconds value must be between 30 and 3600. The default is 60.
This value will be used for all Data Domain virtual libraries managed by this MCP host. This value is not persistent and must be reset any time the DDSUPPORT software is restarted.

NOTE: Changes to this value can affect the responses to other actions in the Data Domain virtual libraries.

To set the timer for checking the tape library vault to 90 seconds, from the ODT enter:
6265AX CONFIGURE VAULT 90
If you then check the ODT messages, you should see:

```
6265 15:54 DDVTL:CONFIGURE VAULT is 90
```

If you don't remember what you set the timer to last and want to find out what the current setting is just enter from the ODT:
6265AX CONFIGURE
Then check the ODT messages (**MSG**):

```
6265 15:57 DDVTL:CONFIGURE VAULT is 90
```

Appendix A
Library Support Programming Interface

The following section describes the programming interface for the Library Support software library. The interface has been designed to be hardware independent.

Caution

DSI reserves the right to change this interface at any time. DSI will attempt to give ample warning of changes where practical. The user should review the release notes included with each release to determine if any changes have been or will be made to this interface.

Declaring the Library Support Library

```
LIBRARY LIBRARYSUPPORT(
                         LIBACCESS=BYFUNCTION
                        ,FUNCTIONNAME="TAPELIBRARYSUPP."
                        );
```

The Library Support software library should always be declared by function. This allows the user to place the object and configuration files on any family or with a usercode. The caller then does not need to program for changes in file locations.

SYSTEM/DSISUPPORT

This file is a support library used by the TapeManager system and other products available from DSI. This library contains common procedures used by these software's. This library must be SLed as DSISUPPORT with the library attributes of ONEONLY and TRUSTED.

Example: SL DSISUPPORT = SYSTEM/DSISUPPORT: TRUSTED, ONEONLY. The library code file is supplied with the CONTROL, PU, LOCKED, and IDENTITY MP commands applied to it. (The library must be MP+ PU at a minimum.)

General Procedure Attributes

Except as noted, all entry points in the Library Support software library have the following characteristics in common.

Element IDs

An ID is used to reference various elements of a library. An ID consists of a word of type REAL. The ID word consists of a number of fields that describe the element. The ID should always be used as a full REAL word. The individual fields within the ID should never be referenced since they are not documented and subject to change.

Strings

Unless otherwise noted, string parameters are always passed or returned in EBCDIC arrays. Strings whether passed or returned should always start in the first character and end with a NUL character. If the EBCDIC array that is passed is not large enough to hold the string being returned it will be resized so that it is large enough to hold the returned string.

Bar Codes

Cartridges used with automated libraries can be used much more efficiently if they have bar code labels attached to them. This software supports bar code labels of 6 characters or more. However, when more than 6 characters are on the label only the lower order 6 characters are used. This is due to the MCP system limitation of 6-character serial numbers. It is highly recommended that the calling software enforce the restriction that the recorded serial number (SN) match that of the bar code label. For that reason, in the following procedures that pass bar code parameters, the bar code parameters must be formatted according to MCP system serial number rules.

Cartridges without bar code labels may be placed in a library. A cartridge without a bar code will have "??????" returned in the bar code (volume tag) field of the inventory and status requests. A cartridge with a damaged bar code (could not be read) will also appear this way. Warning messages will be displayed by the software whenever a cartridge without a bar code label is used.

Libraries that have bar code scanners return the actual scanned value in “VOL_TAG” parameters. Libraries not having a bar code scanner will return the cartridge serial number as read by the MCP system during the inventory process.

Procedure Results

Each procedure is defined as type BOOLEAN. If the procedure completes without error the value returned is FALSE ([00:01] = 0). If there is an error the procedure will return a value of TRUE ([00:01] = 1). When the procedure value is true an error value is returned in bits [26:10] of the procedure value. The following table describes the possible error values.

Value	**Description**
0	No error. The procedure completed correctly.
1	Bad name. A badly constructed name was passed.
2	Not online. The referenced controller or family is not online.
3	Invalid controller. The family name is not valid, or the family specified is not a DSI Controller.
4	Controller limit. The maximum number of controllers allowed was exceeded.
5	Not yet implemented. The requested feature has not been implemented.
6	IO error. An IO error occurred between the MCP system and the DSI Controller. NOTE: This does not necessarily imply an MCP IO error.
7	Not found. The requested item was unavailable or not found.
8	Command error. The library was passed an invalid command.
9	Bad length. A parameter was passed with an invalid length.
10	Slot not empty. An attempt was made to move a cartridge to a slot that was not empty.

11	Door empty. An attempt was made to move a cartridge from a door that is empty.
12	Slot empty. An attempt was made to move a cartridge from a slot that was empty.
13	Door not empty. An attempt was made to move a cartridge to a door that was full.
14	Cartridge in use. An attempt was made to select a cartridge that was being used by a drive unit.
15	Bad Library ID. An invalid library id was passed as a parameter.
16	Bad slot. A slot number was passed that was out of range for the referenced library.
17	Bad unit ID. An invalid unit id was passed as a parameter.
18	Bad time. An invalid time (seconds) parameter was passed.
19	Bad door ID. An invalid door id was passed as a parameter.
20	Bad action. An invalid action parameter was passed.
21	Bad tag. An invalid tag parameter was passed. (DEIMPLEMETED)
22	Not same library. Unit/door/slot/library ids passed to a procedure were not from the same library.
23	Drive in use. For an unload procedure, denotes that the drive has not completed its rewind and eject cycle. For a load procedure, denotes that there is already a cartridge in the unit.
24	Library error. The library reported a hardware error.
25	Library busy. The library reported it was busy with another operation. Retry later.
26	Library offline. The library is offline. This could be due to a DISABLE command, pressing the online/offline button on the library, or opening the library door.
27	Inventory changed. A warning that the library inventory has changed. Only returned by the LIBRARY_COMMAND procedure.
28	Drive empty. An attempt was made to move a cartridge from a drive unit that was empty.
29	Software error. Tape Library Support software error. Most likely an error having to do with a remote library.
30	Port open. The IO port (mailbox) of the library is open so the command could not be completed.
31	Not supported. The library does not support the function. Primarily returned by CSC-A connected libraries.
32	Not VTL. The library has not been declared as a VTL (i.e., the VTL Agent connection declaration) therefore the requested function is not available.
33	No Agent. The VTLSUPPORT library is not running or has not been set up.
34	Not Supported. The function requested is not supported with VTL libraries.
35	Not backend library. The library has not been declared as a VTL back-end library (i.e. the VTL connection declaration) therefore the requested function is not available.
36	Not Licensed. This function is not available as the VTL Agent command functions have not been licensed.

LIBRARY_COMMAND Procedure

```
BOOLEAN PROCEDURE LIBRARY_COMMAND(LIBRARY_ID, COMMAND_LENGTH,
                                  COMMAND, RESPONSE_LENGTH,
                                  RESPONSE);
VALUE LIBRARY_ID, COMMAND_LENGTH;
REAL
     LIBRARY_ID
    ,COMMAND_LENGTH
    ,RESPONSE_LENGTH
    ;
EBCDIC ARRAY
     COMMAND[0]
    ,RESPONSE[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passes a command string to a DSI Controller and returns the response to that command.

Usage: This procedure requires a knowledge of the protocol used to communicate with the DSI Controller. That protocol is not documented here. Therefore, this procedure should not be used without instructions from DSI.

Parameters:

LIBRARY_ID	(Input) the library id of the library that is to process the command.
COMMAND_LENGTH	(Input) the length in characters of the command
COMMAND	(Input) the command in the appropriate format for a DSI Controller
RESPONSE_LENGTH	(Output) length of the response on characters
RESPONSE	(Output) the response from the DSI Controller and/or the library

Results: The DSI Controller and/or library process the command unless an error is detected.

Possible errors: All

LIBRARY_ID_FROM_NAME Procedure

```
BOOLEAN PROCEDURE LIBRARY_ID_FROM_NAME(NAME, LIBRARY_ID);
REAL
     LIBRARY_ID
    ;
EBCDIC ARRAY
     NAME[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed the name of a library this procedure will return the id of that library.

Usage: This procedure is used to determine if a name is a valid library name and to get the id for that library.

Parameters:

NAME	(Input) the name of the library to be searched for. The name must start in the first character of the array and end with a NUL character.
LIBRARY_ID	(Output) the id of the named library if found

Results: The id of the named library is returned.

Possible errors: 7

LIBRARY_ID_FROM_UNIT Procedure

```
BOOLEAN PROCEDURE LIBRARY_ID_FROM_UNIT(UNIT_NUMBER, NAME,
                                       LIBRARY_ID);
VALUE UNIT_NUMBER;
REAL
     UNIT_NUMBER
    ,LIBRARY_ID
    ;
EBCDIC ARRAY
     NAME[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed the unit number of a drive unit that is contained in a library this procedure will return the id of that library and its name.

Usage: This procedure is used to determine if a drive unit is contained within a library.

Parameters:

UNIT_NUMBER	(Input) the unit number of the drive unit to be searched for
NAME	(Output) the name of the library containing the drive unit if found. The name is NUL terminated.
LIBRARY_ID	(Output) the id of the library containing the drive unit if found

Results: The name and id of the library that contains the drive unit is returned.

Possible errors: 7

LIBRARY_NAME Procedure

```
BOOLEAN PROCEDURE LIBRARY_NAME(LIBRARY_ID,NAME);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID
    ;
EBCDIC ARRAY
     NAME[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed the ID of a library this procedure will return the library's name.

Usage: This procedure is used to get a library's name for display in messages and reports.

Parameters:		
	LIBRARY_ID	(Input) the id of the library to be searched for.
	NAME	(Output) the name of the library identified by the id if found. The name is NUL terminated.

Results: The name that is identified by the id is returned.

Possible errors: 15

ALTERNATE_ID_FROM_NAME Procedure

```
BOOLEAN PROCEDURE ALTERNATE_ID_FROM_NAME(NAME, LIBRARY_ID,
                                         ALTERNATE_ID);
REAL
     LIBRARY_ID
    ,ALTERNATE_ID
    ;
EBCDIC ARRAY
     NAME[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed the name of an alternate configuration for a library this procedure will return the id of that alternate library configuration.

Usage: This procedure is used to determine if a name is a valid alternate configuration name and to get the id for that alternate configuration.

Parameters:

NAME	(Input) the name of the alternate configuration to be searched for. The name must start in the first character of the array and end with a NUL character.
LIBRARY_ID	(Input) the id of the library to search for alternates
ALTERNATE_ID	(Output) the id of the alternate library if found

Results: The id of the named alternate configuration is returned.

Possible errors: 7, 15

ALTERNATE_SWITCH Procedure

```
BOOLEAN PROCEDURE ALTERNATE_SWITCH(LIBRARY_ID, ALTERNATE_ID);
VALUE
     LIBRARY_ID
    ,ALTERNATE_ID
    ;
REAL
     LIBRARY_ID
    ,ALTERNATE_ID
    ;
LIBRARY LIBRARYSUPPORT;
```

Function: Changes a library's configuration from the current configuration description to an alternate (or original) configuration description.

Usage: This procedure is used to switch between library configuration descriptions for testing and fail over purposes. The referenced library must already be disabled.

Parameters:

LIBRARY_ID	(Input) the id of the current library configuration.
ALTERNATE_ID	(Output) the id of the alternate library configuration.

Results: The current library configuration is changed to the alternate configuration.

Possible errors: 8, 15

LIBRARY_VERSION Procedure

```
BOOLEAN PROCEDURE LIBRARY_VERSION(LIBRARY_ID, VERSION_INFO);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID
    ;
EBCDIC ARRAY
     VERSION_INFO[0]
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function:	Passed a valid library id, this procedure returns version information regarding the library interface software, the DSI Controller, and the library itself.	
Usage:	This procedure is meant to provide textual information to be displayed to a user requesting software and hardware version information.	
Parameters:	LIBRARY_ID	(Input) the id of the library to return version information for.
	VERSION_INFO	(Output) the version information is returned as a set of strings. Each string is terminated by a NUL character. The last string is terminated by two NUL characters. The number and format of the strings may vary from installation to installation and release to release. Each string will be less than 72 characters.
Results:	Version information of the software and hardware that make up the library system.	

Possible errors: 15, 26

LIBRARY_STATUS Procedure

```
BOOLEAN PROCEDURE LIBRARY_STATUS(LIBRARY_ID, STATUS_VALUE);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID
    ,STATUS_VALUE
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % RETURNED FIELDS IN STATUS_VALUE WORD
     OFF_LINEF        = [0:1]# % 1 = UNIT IS OFFLINE
    ,ACCESS_OPENF     = [1:1]# % 1 = ACCESS DOOR IS OPEN
    ,MAINT_MODEF      = [2:1]# % 1 = UNIT IS IN MAINTENANCE MODE
    ,CONT_OFFLINEF    = [3:1]# % 1 = CONTROLLER OFFLINE (NO COMM)
    ,DISABLEDF        = [4:1]# % 1 = LIBRARY WAS DISABLED
    ,DOING_INVENF     = [5:1]# % 1 = MANUAL INVENTORY IN PROCESS
    ,PORT_NOT_OPENF   = [6:1]# % 1 = CLIENT PORT TO SERVER NOT OPEN
    ,STANDBY_STOPF    = [7:1]# % 1 = STANDBY OR STOP BUTTON PUSHED
    ,ACSLS_LIBF       = [8:1]# % 1 = LIBRARY CONNECTED VIA CSC-A
    ,VIAVTL_LIBF      = [9:1]# % 1 = LIBRARY CONNECTED VIA VTL
    ,NOCONN_LIBF      = [10:1]# % 1 = LIBRARY CONNECTION = NONE
    ,SENSE_KEYF       = [31:8]# % SENSE KEY FROM REQUEST SENSE
    ,ASCF             = [39:8]# % ADDITIONAL SENSE CODE
    ,ASCQF            = [47:8]# % ADDITIONAL SENSE CODE QUALIFIER
    ;
```

Function: Passed a valid library id, this procedure returns the current status information from a library.

Usage: This procedure is meant to provide a real-time status report of the condition of a library.

Parameters:

LIBRARY_ID — (Input) the id of the library to return status information for.

STATUS_VALUE — (Output) the status information is returned as a set of fields in a word. The fields of this word are defined as follows:

[00:01] = 1	if the library is marked offline. The following fields may help determine why the library is off-line.
[01:01] = 1	if the library's access door is open.
[02:01] = 1	if the library has been put in maintenance mode
[03:01] = 1	if communication to the DSI Controller has failed
[04:01] = 1	if the library has been disabled by the LIBRARY_DISABLE procedure
[05:01] = 1	if the manual inventory process for non-barcode libraries is running
[06:01] = 1	if the this is a client library and the port to the server is closed

[07:01] = 1	if the standby or stop button has been detected as having been pressed for this library
[08:01] = 1	if the library is accessed via the CSC-A software
[09:01] = 1	if the library is accessed via the VTL (i.e., it is a VTL back end library)
[10:01] = 1	if the library is declared as CONNECTION = NONE
[23:19]	is reserved for future use
[31:08]	is the sense key value for the library
[39:08]	is the additional sense code (ASC)
[47:08]	is the additional sense qualifier (ASCQ)

The sense key, ASC, and ASCQ fields are provided for display and logging purposes. The values in these fields will vary from library to library. Refer the manufacturers documentation for library specific values.

Results: A status request is made to the library.

Possible errors: 15, 26

LIBRARY_ENABLE Procedure

```
BOOLEAN PROCEDURE LIBRARY_ENABLE(LIBRARY_ID);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) TO ENABLE
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed a valid library id, this procedure marks a library as enabled and attempts to bring a library on-line.

Usage: This procedure is meant to be used after calling the LIBRARY_DISABLE procedure. Whether the library is disabled/off-line or not, this procedure causes a full re-initialization of the communication between the host, the DSI Controller, and the library.

Parameters:	LIBRARY_ID	(Input) the id of the library to be enabled.

Results: The library is marked as enabled. The library is brought back on-line if possible.

Possible errors: 15, 26

LIBRARY_DISABLE Procedure

```
BOOLEAN PROCEDURE LIBRARY_DISABLE(LIBRARY_ID);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) TO DISABLE
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed a valid library id, this procedure marks a library as disabled. A disabled library returns an off-line error for all attempts to access it other than enable, status, and info.

Usage: This procedure allows communication to be stopped with a library. Disabling a library allows it to be worked on without causing errors at the host system.

Parameters: LIBRARY_ID (Input) the id of the library to be disabled.

Results: The library is marked as disabled. The host and the DSI Controller will no longer attempt to communicate with this library. If the library is a remote library, the communication to the remote host is severed but the library is not disabled on the remote host.

Possible errors: 15, 6

LIBRARY_INFO Procedure

```
BOOLEAN PROCEDURE LIBRARY_INFO(LIBRARY_ID, LIBRARY_DATA);
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) TO RETURN INFO FOR
    ;
EBCDIC ARRAY
     LIBRARY_DATA[0,0]  % LIBRARY DATA RETURNED
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % LAYOUT OF LIB_DATA ARRAY RETURNED
     LIBRARY_TITLE        = LIB_DATAW[0]# % FOR 3 WORDS
    ,LIBRARY_ALTERNATES   = LIB_DATAW[3].[47:4]#
    ,LIB_DATA_VER         = LIB_DATAW[3].[43:4]#% FORMAT VERSION
    ,LIBRARY_NOCONN       = LIB_DATAW[3].[23:1]#
    ,LIBRARY_DD           = LIB_DATAW[3].[22:1]#
    ,LIBRARY_VIAVTL       = LIB_DATAW[3].[21:1]#
    ,LIBRARY_VTL          = LIB_DATAW[3].[20:1]#
    ,LIBRARY_ACSLS        = LIB_DATAW[3].[19:1]#
    ,LIBRARY_CSCA         = LIB_DATAW[3].[18:1]#
    ,LIBRARY_ALTERNATE    = LIB_DATAW[3].[17:1]#
    ,LIBRARY_IS_MO        = LIB_DATAW[3].[15:1]#
    ,LIBRARY_IS_CD        = LIB_DATAW[3].[14:1]#
    ,LIBRARY_IMPORT_SUPPORT=LIB_DATAW[3].[13:1]#
    ,LIBRARY_IMPORT_AUTO  = LIB_DATAW[3].[12:1]#
    ,LIBRARY_IMPORT_COUNT = LIB_DATAW[3].[11:1]#
    ,LIBRARY_MEDIA_2SIDED = LIB_DATAW[3].[10:1]#
    ,LIBRARY_REMOTE       = LIB_DATAW[3].[9:1]#
    ,LIBRARY_NO_BARCODE   = LIB_DATAW[3].[8:1]#
    ,LIBRARY_TYPE         = LIB_DATAW[3].[7:8]#
    ,LIBRARY_CONNECTION   = LIB_DATAW[4]#
    ,LIBRARY_CONN_TYPE    = LIB_CONNECTION.[47:8]#
    ,LIBRARY_CONN_PORTTYPE= LIB_CONNECTION.[23:8]#
    ,LIBRARY_CONN_PORTID  = LIB_CONNECTION.[15:16]#
    ,LIBRARY_CONN_NAME    = LIB_DATAW[5]# % FOR 3 WORDS
    ,LIBRARY_SLOTS        = SLOT_DATAW[0]#
    ,LIBRARY_SLOTS_FIRST  = SLOT_DATAW[1]#
    ,LIBRARY_SLOTS_LAST   = SLOT_DATAW[2]#
    ,LIBRARY_DOORS        = DOOR_DATAW[0]#
    ,LIBRARY_DOOR_TYPE(I) = DOOR_DATAW[((I-1)*2)+1].[47:8]#
    ,LIBRARY_DOOR_CAP(I)  = DOOR_DATAW[((I-1)*2)+1].[39:8]#
    ,LIBRARY_DOOR_IO(I)   = DOOR_DATAW[((I-1)*2)+1].[31:8]#
    ,LIBRARY_DOOR_ENMBR(I)= DOOR_DATAW[((I-1)*2)+2]#
    ,LIBRARY_DRIVES       = DRIVE_DATAW[0]#
    ,LIBRARY_DRIVE_TYPE(I)= DRIVE_DATAW[((I-1)*2)+1].[47:8]#
    ,LIBRARY_DRIVE_NUM(I) = DRIVE_DATAW[((I-1)*2)+1].[39:16]#
    ,LIBRARY_DRIVE_ADDL(I)= DRIVE_DATAW[((I-1)*2)+1].[7:8]#
    ,LIBRARY_DRIVE_ENMBR(I)=DRIVE_DATAW[((I-1)*2)+2]#
    ,LIBRARY_DRIVE_SHARED(I)=
            DRIVE_DATAW[LIBRARY_DRIVE_ADDL(I)].[0:1]#
    ,LIBRARY_DRIVE_SHARED_UNIT(I)=
            DRIVE_DATAW[LIBRARY_DRIVE_ADDL(I)].[39:16]#
    ,LIBRARY_DRIVE_SHARED_HOST_SIZE(I)=
            DRIVE_DATAW[LIBRARY_DRIVE_ADDL(I)+1].[47:8]#
    ,LIBRARY_DRIVE_SHARED_HOST(I)=
            POINTER(DRIVE_DATAW[LIBRARY_DRIVE_ADDL(I)+1])+1#
    ;

ARRAY LIB_DATAW[0]   = LIB_DATA[0,*];
ARRAY SLOT_DATAW[0]  = LIB_DATA[1,*];
ARRAY DOOR_DATAW[0]  = LIB_DATA[2,*];
ARRAY DRIVE_DATAW[0] = LIB_DATA[3,*];
```

Function: Passed a valid library id, this procedure returns static configuration information for a library.

Usage: This procedure returns library configuration information as found in the SYSTEM/TAPELIBRARY/CONFIGURATION file and returned when interrogating the library when first connecting. This information is static and does not change during an execution of the library software.

Parameters:

LIBRARY_ID	(Input/Output) the id of the library whose configuration is to be returned. The library id parameter of this procedure call has a special feature that allows the caller to pass a simple integer and the ID will be changed to a valid library id if that library exists. The intent is to pass the integers 1 to N until the "not found" (7) error is returned. (Note: Intermediate "bad library id" (15) may occur.) This allows the caller to find out the number and IDs of any connected libraries.
LIBRARY_DATA	(Output) this is a multi-dimension array with a row for the library, a row for its slots, a row for its doors, and a row for its drives. It is easiest to use if this array is row equated as in the example.

Results: The library configuration is returned.

Possible errors: 7, 15

LIBRARY_INVENTORY Procedure

```
BOOLEAN PROCEDURE LIBRARY_INVENTORY(LIBRARY_ID, SLOT, INVEN_DATA);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) TO RETURN INFO FOR
    ,SLOT               % SLOT # RETURN INFO FOR (< 0 = ALL SLOTS)
    ;
EBCDIC ARRAY
     INVEN_DATA[0]      % INVENTORY DATA RETURNED
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % LAYOUT OF INVEN_DATA ARRAY RETURNED
     SLOT_NUMBER(I)      = INVEN_DATAW[((I-1)*2)+0].[47:16]#
    ,SLOT_ACCESS(I)      = INVEN_DATAW[((I-1)*2)+0].[30: 1]#
    ,SLOT_FULL(I)        = INVEN_DATAW[((I-1)*2)+0].[28: 1]#
    ,SLOT_2SIDED(I)      = INVEN_DATAW[((I-1)*2)+0].[27: 1]#
    ,SLOT_NO_BARCODE(I)  = INVEN_DATAW[((I-1)*2)+0].[26: 1]#
    ,SLOT_DRIVE_SIDE2(I)= INVEN_DATAW[((I-1)*2)+0].[25: 1]#
    ,SLOT_DRIVE(I)       = INVEN_DATAW[((I-1)*2)+0].[15:16]#
    ,SLOT_VOL_TAG(I)     = INVEN_DATAW[((I-1)*2)+1]#
    ;

ARRAY INVEN_DATAW[0] = INVEN_DATA;
```

Function: This procedure returns the current status of one or all of the storage slots within a library.

Usage: This procedure allows the caller to determine which storage slots hold data cartridges and the bar code labels of the cartridges. This information should not be stored since it may change dynamically. Used primarily for status and reporting purposes.

Parameters:

LIBRARY_ID	(Input) the id of the library that contains the storage slot(s).
SLOT	(Input) slot number to return inventory information for. If the value passed is less than 0 (negative), then inventory information for all slots is returned. Passing the value -99 will return the inventory of the virtual vault if the LIBRARY_ID refers to a virtual library.
INVEN_DATA	(Output) 2 words of information are returned for each slot. In the first word;

Word 0

[47:16]	the slot number,
[30:01] = 1	if the slot is usable,
[28:01] = 1	if the slot has a cartridge assigned to it (Note: cartridge may be in a drive)
[27:01] = 1	if the slot contains 2 sided media
[26:01] = 1	if the library does not have a bar code reader (Note: this is not the same as the media not having a bar code.)

[25:01] = 1	if the slot is in use (in a drive unit) and side 2 of 2-sided media is in use
[15:16]	if the cartridge assigned to this slot is in use, the unit number of the drive that is using the cartridge.

All other fields in this word are reserved for future expansion.

Word 1

The second word contains the low order 6 characters of the cartridges bar code label.

Results: The current slot status information is returned. The library status was checked.

Possible errors: 6, 15, 16, 24, 25, 26

LIBRARY_INITIALIZE Procedure

```
BOOLEAN PROCEDURE LIBRARY_DISABLE(LIBRARY_ID, WHAT);
VALUE LIBRARY_ID, WHAT;
REAL
     LIBRARY_ID          % LIBRARY ID (HANDLE) TO DISABLE
    ,WHAT                % INITIALIZE OPTION (NOT CURRENTLY USED)
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Passed a valid library id, this procedure causes a cartridge library to perform an initialize element status for its storage slots. For some libraries this function only determines which storage slots are full, other libraries will also scan the cartridge bar codes if so equipped.

Usage: This procedure is intended for use with libraries that do not automatically do a initialize element status when the bulk load door has been opened. While this command will generally work with other libraries, it is redundant and not needed for libraries that do this function automatically.

Parameters:

LIBRARY_ID	(Input) the id of the library to be initialized
WHAT	(Input) initialize option (not currently used)

Results: The library’s inventory is updated. For libraries without barcode readers, a manual inventory of all slots is started. **NOTE**: *Depending on the size and type of the cartridge library, this procedure can take a significant amount of time (minutes) to complete*.

Possible errors: 15, 26

LIBRARY_DISPLAY Procedure

```
BOOLEAN PROCEDURE LIBRARY_DISPLAY(UNIT_ID, SECONDS, MSG,
                                  MSG_LENGTH);
VALUE UNIT_ID, SECONDS, MSG_LENGTH;
REAL
     UNIT_ID            % TAPE UNIT TO DISPLAY MESSAGE ON/FOR
    ,SECONDS            % TIME IN SECONDS TO DISPLAY MESSAGE
    ,MSG_LENGTH         % LENGTH OF MESSAGE IN CHARACTERS
    ;
EBCDIC ARRAY
     MSG[0]             % MESSAGE TO BE DISPLAYED
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Places a message in the display panel of the library.

Usage: Many libraries have a programmable display panel. For libraries with multiple lines of display, a line is assigned to each possible drive within the library. This procedure is meant to display operator information such as the label or serial number of the cartridge in a drive.

Parameters:

UNIT_ID	(Input) the id of drive unit to display the message for.
SECONDS	(Input) the number of seconds to display the message. Zero seconds displays the message until changed by another command. (Note: many libraries do not support a timed display, for these libraries seconds is always treated as zero.)
MSG	(Input) the message to be displayed in EBCDIC
MSG_LENGTH	(Input) the number of characters in the MSG parameter to display. Must be 20 or less. If the model of library does not support 20 character displays, the right most characters are truncated to make the message fit.

Results: The message is displayed for the drive unit. NOTE: some libraries do not support a programmable display. For these libraries the call is ignored, and no error is returned.

Possible errors: 9, 17, 26

LIBRARY_LOG Procedure

```
BOOLEAN PROCEDURE LIBRARY_LOG(LIBRARY_ID, LOG_INFO);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) TO RETURN INFO FOR
    ;
EBCDIC ARRAY
     LOG_INFO[0]        % LOG MESSAGES RETURNED
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Returns any log messages generated by the Library Controller.

Usage: When notification is received from the Library Controller that it has log messages to send, this procedure is called to retrieve them from the Library Controller.

Parameters:	LIBRARY_ID	(Input) the ID of the library that is controlled by the Library Controller.
	LOG_INFO	(Output) an array containing text lines delimited by CR and LF characters. Last line terminated with a NUL.

Results: Any stored Library Controller messages are returned.

Possible errors: 9,15

PICKER_ID_FROM_LIBRARY Procedure

```
BOOLEAN PROCEDURE PICKER_ID_FROM_LIBRARY(LIBRARY_ID,PICKER,
                                         PICKER_ID);
VALUE LIBRARY_ID, PICKER;
REAL
     LIBRARY_ID         % ID OF LIBRARY PICKER IS IN
    ,PICKER             % PICKER NUMBER
    ,PICKER_ID          % RETURNED IDENTIFIER FOR PICKER
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a library picker (also known as gripper or robot) referenced by the picker number and returns its ID.

Usage: This procedure will return the picker ID for the picker referenced by the picker number. The results of this procedure are normally used with the LIBRARY_MOVE procedure.

Parameters:

	LIBRARY_ID	(Input) the ID of the library that the picker is in.
	PICKER	(Input) the number of the picker to be referenced.
	PICKER_ID	(Output) the ID of the picker number referenced.

Results: The library information array is searched for the picker with the matching picker number in the library, returning its ID.

Possible errors: 7

PASSTHRU_ID_FROM_LIBRARY Procedure

```
BOOLEAN PROCEDURE PASSTHRU_ID_FROM_LIBRARY(LIBRARY_ID,PASSTHRU_ID);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % ID OF LIBRARY PASSTHRU IS IN
    ,PASSTHRU_ID        % RETURNED IDENTIFIER FOR PASSTHRU
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a library passthru device in the library and returns its ID.

Usage: This procedure will return the passthru ID for the referenced library if the library has one. The results of this procedure are normally used with the LIBRARY_MOVE procedure.

Parameters:

LIBRARY_ID	(Input) the ID of the library that the passthru is in.
PASSTHRU_ID	(Output) the ID of the passthru returned.

Results: The library information array is searched for the passthru in the library (if any), returning its ID.

Possible errors: 7

DOOR_ID_FROM_NAME Procedure

```
BOOLEAN PROCEDURE DOOR_ID_FROM_NAME(DOOR_NAME,LIBRARY_ID,DOOR_ID);
REAL
     LIBRARY_ID         % ID OF LIBRARY DOOR IS ATTACHED TO
    ,DOOR_ID            % RETURNED IDENTIFIER FOR DOOR
    ;
EBCDIC ARRAY
     DOOR_NAME[0]       % NAME OF DOOR TO SEARCH FOR
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a library door by name and returns its ID and the ID of the library it is attached to.

Usage: This procedure will return the door and library IDs for the named door. Each door in a library configuration must have a unique name. This procedure is useful in command parsing to determine the validity and ID of a door used in a command.

Parameters:

DOOR_NAME	(Input) name of the door to be searched for terminated by a null.
LIBRARY_ID	(Output) the ID of the library that the door is attached to.
DOOR_ID	(Output) the ID of the named door.

Results: The door information array is searched for the named door, returning its ID and the ID of the library it is attached to.

Possible errors: 7

DOOR_ID_FROM_ELEMENT Procedure

```
BOOLEAN PROCEDURE DOOR_ID_FROM_ELEMENT(ELEMENT,LIBRARY_ID,DOOR_ID);
VALUE ELEMENT, LIBRARY_ID;
REAL
     ELEMENT            % LIBRARY'S ELEMENT NUMBER FOR DOOR
    ,LIBRARY_ID         % ID OF LIBRARY DOOR IS ATTACHED TO
    ,DOOR_ID            % RETURNED IDENTIFIER FOR DOOR
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a library door referenced by the element number and returns its ID.

Usage: This procedure will return the door ID for the door referenced by the element number. This procedure is used primarily for diagnostics.

Parameters:

ELEMENT	(Input) the element number of the door to be searched for.
LIBRARY_ID	(Input) the ID of the library that the door is attached to.
DOOR_ID	(Output) the ID of the door that has the element number.

Results: The door information array is searched for the door with the matching element number in the library, returning its ID.

Possible errors: 7

DOOR_NAME Procedure

```
BOOLEAN PROCEDURE DOOR_NAME(DOOR_ID,NAME);
VALUE DOOR_ID;
REAL
     DOOR_ID            % DOOR ID (HANDLE) TO RETURN NAME OF
    ;
EBCDIC ARRAY
     NAME[0]            % NAME OF DOOR (NULL TERMINATED)
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Returns the name of a door referenced by the ID.

Usage: This procedure is used to find a name given a door ID. It is used primarily in building error and response messages to user.

Parameters:

DOOR_ID	(Input) the ID of the door to return the name for.
NAME	(Output) the name of the door referenced by the ID. The name is terminated by a null. If the passed array is too small to hold the name it will be resized so that it will hold the complete name.

Results: The door name is returned.

Possible errors: 19

DOOR_STATUS Procedure

```
BOOLEAN PROCEDURE DOOR_STATUS(DOOR_ID, STATUS_DATA);
VALUE DOOR_ID;
REAL
     DOOR_ID            % DOOR ID (HANDLE) TO RETURN STATUS FOR
    ;
EBCDIC ARRAY
     STATUS_DATA[0]     % STATUS INFO RETURNED FOR DOOR
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % LAYOUT OF STATUS_DATA ARRAY RETURNED FROM DOOR_STATUS
     DOOR_ENMBR     = STATUS_DATAW[0].[47:16]#
    ,DOOR_SLOTNUM   = STATUS_DATAW[0].[31:16]#
    ,DOOR_DD        = STATUS_DATAW[0].[11:1]# % DOOR IN DATA DOMAIN
    ,DOOR_VIAVTL    = STATUS_DATAW[0].[10:1]# % DOOR IN BACKEND LIB
    ,DOOR_VTL       = STATUS_DATAW[0].[9 :1]# % DOOR IN VTL
    ,DOOR_LA        = STATUS_DATAW[0].[8: 1]# % LIBATTCH CONTROLLED
    ,DOOR_CSCA      = STATUS_DATAW[0].[7: 1]# % CSC-A CONTROLLED
    ,DOOR_INOUT     = STATUS_DATAW[0].[6: 2]# % IN,OUT,IO
    ,DOOR_FROM_OPER = STATUS_DATAW[0].[4: 1]# % OPER PUT IN
    ,DOOR_ACCESS    = STATUS_DATAW[0].[3: 1]# % ACCESS O.K.
    ,DOOR_EXCEPT    = STATUS_DATAW[0].[2: 1]#
    ,DOOR_FULL      = STATUS_DATAW[0].[0: 1]# % DOOR HAS TAPE
    ,DOOR_VOL_TAG   = STATUS_DATAW[1]#        % BAR CODE LABEL
;

ARRAY STATUS_DATAW[0] = STATUS_DATA;
```

Function: Returns current status information about a library door.

Usage: This procedure will return the current status information for a library door. The procedure is used to diagnose errors and create information displays for the user.

Parameters:

DOOR_ID (Input) the ID of the door to return status information for.

STATUS_DATA (Output) the status information is returned as a set of fields. These fields are defined as follows:

Word 0

[47:16]	the door element number assigned by the library
[31:16]	if the door received a cartridge from a slot, this is the sending slot number
[11:01] = 1	if the door is in a Data Domain VTL
[10:01] = 1	if the door is in a back-end library
[09:01] = 1	if the door is part of a virtual library
[08:01] = 1	if the door is controlled by StorageTek Library Attach.
[07:01] = 1	if the door is controlled by Unisys CSC-A.

[06:02]	same as VALUE(IN, OUT, or IO). Operation door is capable of.
[04:01] = 1	if cartridge was placed in door by operator (Note: not reported by all libraries.)
[03:01] = 1	if the door is accessible (can be used).
[02:01] = 1	if the door has an error reported against it.
[00:01] = 1	if the door has a cartridge in it.

All other fields in this word are reserved for future use.

Word 1

If the door has a cartridge in it, this field contains the right most 6 characters of the bar code label of the cartridge.

Results: Information showing the current status of the referenced door is returned. The door status is checked.

Possible errors: 19

LIBRARY_MOVE Procedure

```
BOOLEAN PROCEDURE LIBRARY_MOVE(SOURCE_ID, DESTINATION_ID);
VALUE SOURCE_ID, DESTINATION_ID;
REAL
     SOURCE_ID          % ID OF UNIT/DOOR/SLOT TO GET TAPE FROM
    ,DESTINATION_ID     % ID OF UNIT/DOOR/SLOT TO PLACE TAPE IN
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Moves a cartridge from source location to destination location.

Usage: This procedure is a generic cartridge movement routine. Very little error recovery is provided and must be handled by the caller. The source and destination ID must refer to the same library. NOTE: depending on the library hardware and the source/destination combination the return of this procedure may take from 10 seconds to 2 minutes.

Parameters:	SOURCE_ID	(Input) the ID of the door, drive, or slot to get the cartridge from.
	DESTINATION_ID	(Input) the ID of the door, drive, or slot to send the cartridge to.

Results: The cartridge is attempted to be moved from the source location to the destination location. Inventory data is updated.

Possible errors: 14, 17, 19, 22, 23, 28

LIBRARY_MATCH Procedure

```
BOOLEAN PROCEDURE LIBRARY_MATCH(ID1, ID2);
VALUE ID, LIBRARY_ID;
REAL
     ID1
    ,ID2
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Checks if two IDs belong to the same library.

Usage: This procedure is used to check if two elements belong to the same library system such as if a drive unit is in a library.

Parameters:

ID1	(Input) the ID of a door, slot, drive, or library to be checked against ID2.
ID2	(Input) the ID of a door, slot, drive, or library to be checked against ID1.

Results: The procedure returns a TRUE or FALSE result.

Possible errors: 22

UNIT_ID_FROM_UNIT Procedure

```
BOOLEAN PROCEDURE UNIT_ID_FROM_UNIT(UNIT_NUMBER,UNIT_ID,
                                    LIBRARY_ID);
VALUE UNIT_NUMBER;
REAL
     UNIT_NUMBER        % MCP EXTERNAL UNIT NUMBER
    ,UNIT_ID            % RETURNED UNIT ID
    ,LIBRARY_ID         % RETURNED ID OF LIBRARY UNIT ATTACHED TO
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a drive unit with the specified unit number and returns that drives ID and the ID of the library it is attached to.

Usage: This procedure is used to find the ID of a drive referenced by unit number. This procedure is useful when parsing commands and determining if a drive is a library drive when analyzing log information.

Parameters:

UNIT_NUMBER	(Input) the MCP unit number of the drive.
UNIT_ID	(Output) the ID of drive if found.
LIBRARY_ID	(Output) the ID of the library the drive is attached to.

Results: The procedure returns the drive unit and library IDs for that unit number.

Possible errors: 7

UNIT_ID_FROM_ELEMENT Procedure

```
BOOLEAN PROCEDURE UNIT_ID_FROM_ELEMENT(LIBRARY_ID,ELEMENT,
                                       UNIT_ID);
VALUE LIBRARY_ID, ELEMENT;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) THAT HAS UNIT
    ,ELEMENT            % LIBRARY'S ELEMENT NUMBER FOR UNIT
    ,UNIT_ID            % RETURNED UNIT ID
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for a library drive unit referenced by the element number and returns its ID.

Usage: This procedure will return the drive unit ID for the drive referenced by the element number in the designated library. This procedure is used primarily for diagnostics.

Parameters:

LIBRARY_ID	(Input) the ID of the library that the drive unit is attached to.
ELEMENT	(Input) the element number of the drive unit to be searched for.
UNIT_ID	(Output) the ID of the drive unit that has the element number.

Results: The unit information array is searched for the drive unit with a matching element in the library, returning its ID.

Possible errors: 7, 15

UNIT_NUMBER Procedure

```
INTEGER PROCEDURE UNIT_NUMBER(UNIT_ID);
VALUE UNIT_ID;
REAL
     UNIT_ID           % UNIT ID (HANDLE) TO RETURN UNIT NUMBER OF
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: For a drive unit ID, returns that unit’s MCP unit number.

Usage: This procedure will return the drive unit's MCP unit number. This procedure is useful when building response and error messages.

Parameters: UNIT_NUMBER (Input) the ID of the drive unit to returns is unit number.

Results: Returns the drive unit number as the procedure value if the ID is valid. If the ID is not valid zero is returned.

Possible errors: none

UNIT_STATUS Procedure

```
BOOLEAN PROCEDURE UNIT_STATUS(UNIT_ID, STATUS_DATA);
VALUE UNIT_ID;
REAL
     UNIT_ID            % UNIT ID (HANDLE) TO RETURN STATUS FOR
    ;
EBCDIC ARRAY
     STATUS_DATA[0]     % STATUS INFO RETURNED FOR UNIT
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % LAYOUT OF STATUS_DATA ARRAY RETURNED FROM UNIT_STATUS
     UNIT_EXT_NUMBER   = STATUS_DATAW[0].[47:16]#
    ,UNIT_SLOTNUM      = STATUS_DATAW[0].[31:16]#
    ,UNIT_DD           = STATUS_DATAW[0].[12:1]# % UNIT IN DD VTL
    ,UNIT_VIAVTL       = STATUS_DATAW[0].[11:1]# % UNIT IN BACKEND
    ,UNIT_VTL          = STATUS_DATAW[0].[10:1]# % UNIT IN VTL LIB
    ,UNIT_LA           = STATUS_DATAW[0].[9: 1]# % LIBAT CONTROLLED
    ,UNIT_CSCA         = STATUS_DATAW[0].[8: 1]# % CSC-A CONTROLLED
    ,UNIT_RESV_PENDING = STATUS_DATAW[0].[7: 1]# % RESERVE PENDING
    ,UNIT_2SIDED       = STATUS_DATAW[0].[6: 1]# % 2 SIDED MEDIA
    ,UNIT_NO_BARCODE   = STATUS_DATAW[0].[5: 1]# % NO BARCODE RDR
    ,UNIT_SIDE2        = STATUS_DATAW[0].[4: 1]# % SIDE 2 LOADED
    ,UNIT_ACCESS       = STATUS_DATAW[0].[3: 1]# % ACCESS ALLOWED
    ,UNIT_EXCEPT       = STATUS_DATAW[0].[2: 1]# % ABNORMAL STATE
    ,UNIT_RESERVED     = STATUS_DATAW[0].[1: 1]# % UNIT RESERVED
    ,UNIT_FULL         = STATUS_DATAW[0].[0: 1]# % UNIT HAS A CART
    ,UNIT_VOL_TAG      = STATUS_DATAW[1]#        % BAR CODE SIDE 1
    ,UNIT_VOL_TAG2     = STATUS_DATAW[2]#        % BAR CODE SIDE 2
    ,UNIT_RESERVE_REASON=STATUS_DATAW[3]#        % RESERVE REASON
    ;

ARRAY STATUS_DATAW[0] = STATUS_DATA;
```

Function: Returns current status information about a library drive unit.

Usage: This procedure will return the current status information for a library drive unit. The procedure is used to diagnose errors and create information displays for the user.

Parameters:

UNIT_ID	(Input) the ID of the drive unit to return status information for.
STATUS_DATA	(Output) the status information is returned as a set of fields. These fields are defined as follows:

Word 0

[47:16]	the MCP unit number of the drive
[31:16]	if the drive received a cartridge from a slot, this is the sending slot number
[12:01] = 1	if the drive is in a Data Domain VTL
[11:01] = 1	if the drive is in a back-end library
[10:01] = 1	if the drive is in a virtual library
[09:01] = 1	if the drive is controlled by StorageTek Library Attach

[08:01] = 1	if the drive is controlled by Unisys CSC-A
[07:01] = 1	if the drive will be reserved when empty
[06:01] = 1	if the drive can use 2-sided media
[05:01] = 1	if the library does not have a bar code reader (Note: this is not the same as the media not having a bar code.)
[04:01] = 1	if the drive is full and side 2 of 2 sided media is loaded
[03:01] = 1	if the drive is accessible (can be used). (NOTE: this is the libraries status for the drive and may not match the MCP host status.)
[02:01] = 1	if the drive has an error reported against it.
[01:01] = 1	if the drive has been reserved.
[00:01] = 1	if the drive has a cartridge in it.

All other fields in this word are reserved for future use.

Word 1

If the drive has a cartridge in it, this field contains the right most 6 characters of the bar code label of the cartridge. If the library does not have a bar code reader, this is the serial number (SN) of the media.

Word 2

If the drive has a cartridge in it, and the media has 2 sides, this is the serial number (SN) of the media’s second side.

Word 3

If the drive is reserved or reserve pending, this is the reason value given for the reserve request.

Results: Information showing the current status of the referenced drive unit is returned. The drive status is checked.

Possible errors: 17

UNIT_MOUNT Procedure

```
PROCEDURE UNIT_MOUNT(UNIT_NUMBER, SN, SNED);
VALUE UNIT_NUMBER, SN, SNED;
REAL
     UNIT_NUMBER        % UNIT NUMBER WHERE CART MOUNTED
    ,SN                 % SERIAL NUMBER OF MOUNTED CART
    ;
BOOLEAN
     SNED               % MOUNT CALL DUE TO SN
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure is called to notify the LibraryManager whenever a cartridge drive mount is seen by the MCP host.

Usage: This procedure is required for support of libraries that do not have bar code readers. When a cartridge is mounted or SNed, the cartridge management system must use this procedure to call LibraryManager so that LibraryManager’ s inventory tables are updated since LibraryManager only recognizes cartridges by their bar code label.

Parameters:

UNIT_NUMBER	(Input) the unit number of drive unit where cartridge was mounted
SN	(Input) the serial number (SN) of the cartridge that was mounted
SNED	(Input) true if the mount call is being made due to the cartridge being SNed

Results: Inventory data is updated.

Possible errors: None

UNIT_LOAD Procedure

```
BOOLEAN PROCEDURE UNIT_LOAD(UNIT_ID, SLOT);
VALUE UNIT_ID, SLOT;
REAL
     UNIT_ID            % UNIT ID (HANDLE) TO BE LOADED
    ,SLOT               % SLOT TO GET CARTRIDGE FROM
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure takes a cartridge from a slot and loads it into a drive unit.

Usage: This procedure will load a drive unit from a slot. It is different from the LIBRARY_MOVE procedure in that it has some additional validity checking and error recovery. NOTE: depending on the library hardware the return of this procedure could take from 10 seconds to 2 minutes. Once this procedure returns it may still some time (drive hardware dependent) until the drive is seen ready by the MCP.

Parameters:

UNIT_ID	(Input) the ID of the drive unit to be loaded.
SLOT	(Input) the slot number to take the cartridge from to load into the drive unit

Results: The drive unit is loaded with the cartridge from the slot. Inventory data is updated.

Possible errors: 12, 14, 16, 17, 23, 26

UNIT_UNLOAD Procedure

```
BOOLEAN PROCEDURE UNIT_UNLOAD(UNIT_ID, SLOT);
VALUE UNIT_ID;
REAL
     UNIT_ID           % UNIT ID (HANDLE) TO BE UNLOADED
    ,SLOT              % SLOT TO SEND CART TO (0 = ORIGINAL SLOT)
    ;                  % RETURNS SLOT TAPE WAS SENT TO
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure takes a cartridge from a drive unit and places it in a slot.

Usage: This procedure will unload a drive unit to a slot. It is different from the LIBRARY_MOVE procedure in that it has some additional validity checking and error recovery. The caller must have issued or seen the drive unit rewind and eject the cartridge before calling this procedure. NOTE: depending on library hardware and drive hardware type this procedure may take from 10 seconds to 2 minutes to return. If this procedure can not complete the unload operation in 2 minutes, a library busy error (25) is returned. The caller should check the MCP status of the drive to determine if the operation should be retried.

Parameters:	UNIT_ID	(Input) the ID of the drive unit to be unloaded.
	SLOT	(Input/Output) the slot number to place the cartridge in. If the value is zero, the cartridge is placed in its original slot. If for any reason, the specified slot is in use, any empty slot will be used. The resulting destination slot is returned.

Results: The drive unit is unloaded and the cartridge placed in a slot. Inventory data is updated.

Possible errors: 16, 17, 23, 26, 28

UNIT_RESERVE Procedure

```
BOOLEAN PROCEDURE UNIT_RESERVE(UNIT_ID, REASON, RESERVE);
VALUE UNIT_ID;
REAL
     UNIT_ID            % UNIT ID (HANDLE) TO BE RESERVED
    ,REASON             % REASON FOR RESERVE REQUEST
    ;
BOOLEAN
     RESERVE            % TRUE = RESERVE, FALSE = UNRESERVE
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure reserves or unreserves a drive unit. A reserved drive unit will not be selected for tape usage.

Usage: This procedure will reserve or unreserve a drive to prevent or allow the drive unit's selection for use. A drive unit that is in use when UNIT_RESERVE is called to reserve a unit will be marked as Reserve Pending and become reserved once the drive is no longer in use. This procedure is used to hold a drive for some special purpose such as cleaning.

Parameters:

UNIT_ID	(Input) the ID of the drive unit to be reserved.
REASON	(Input) integer value denoting reason for reserve request
RESERVE	(Input) If TRUE, reserve the unit, if FALSE unreserve the unit (Output) If TRUE, the unit is reserved, if FALSE the unit is not reserved. If FALSE on a reserve request, [1:1] will be TRUE if the reserve is pending.

Results: The drive unit is reserved, unreserved, or reserve pending.

Possible errors: 17

SLOT_ID_FROM_SLOT Procedure

```
BOOLEAN PROCEDURE SLOT_ID_FROM_SLOT(LIBRARY_ID,SLOT,SLOT_ID);
VALUE LIBRARY_ID, SLOT;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) THAT SLOT IS IN
    ,SLOT               % SLOT NUMBER IN LIBRARY TO RETURN ID FOR
    ,SLOT_ID            % RETURNED SLOT ID (HANDLE)
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for the referenced slot in the library and returns its ID.

Usage: This procedure will return the slot ID for the slot referenced in the designated library. This procedure is used to validate slot numbers in command input and obtain slot IDs for use in other commands.

Parameters:	LIBRARY_ID	(Input) the ID of the library that the slot is in.
	SLOT	(Input) the slot number to return the ID of.
	SLOT_ID	(Output) the ID of the slot.

Results: A slot ID is returned.

Possible errors: 7, 15, 16

SLOT_FROM_SN Procedure

```
BOOLEAN PROCEDURE SLOT_FROM_SN(BARCODE,LIBRARY_ID,SLOT);
VALUE BARCODE;
REAL
     BARCODE          % BAR CODE OF TAPE WE ARE LOOKING FOR
    ,LIBRARY_ID       % RETURNED LIBRARY ID (HANDLE) THAT HAS TAPE
    ,SLOT             % RETURNED SLOT NUMBER IN LIBRARY
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: Searches for the referenced bar code in all libraries and returns the library ID and slot number where the cartridge is found.

Usage: This procedure will return the slot number and library ID where a cartridge can be found. This procedure is used to determine if a specific cartridge is in a library and to get its address (slot number and library ID) for use in other procedures.

Parameters:

BARCODE	(Input) the bar code of the cartridge to be searched for in MCP serial number format.
LIBRARY_ID	(Output) the ID of the library where the cartridge was found.
SLOT	(Output) the slot number where the cartridge was found.

Results: If the requested cartridge is in a library, the cartridge's slot number and library ID are returned. The library status is checked.

Possible errors: 2, 7

SLOT_STATUS Procedure

```
BOOLEAN PROCEDURE SLOT_STATUS(LIBRARY_ID, SLOT, STATUS_DATA);
VALUE LIBRARY_ID, SLOT;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) THAT HAS SLOT
    ,SLOT               % SLOT TO RETURN STATUS FOR
    ;
EBCDIC ARRAY
     STATUS_DATA[0]     % STATUS INFO RETURNED FOR SLOT
    ;
     LIBRARY LIBRARYSUPPORT;

DEFINE % LAYOUT OF STATUS_DATA ARRAY RETURNED BY SLOT_STATUS
     SLOT_STAT_ENMBR    = STATUS_DATAW[0].[47:16]# % ELEMENT NUM
    ,SLOT_STAT_SLOTNUM  = STATUS_DATAW[0].[31:16]# % EXTERNAL NUM
    ,SLOT_STAT_DD       = STATUS_DATAW[0].[8: 1]# % SLOT IN DD VTL
    ,SLOT_STAT_VIAVTL   = STATUS_DATAW[0].[7: 1]# % SLOT IN BACKEND
    ,SLOT_STAT_VTL      = STATUS_DATAW[0].[6: 1]# % SLOT IN VIRTUAL
    ,SLOT_STAT_LA       = STATUS_DATAW[0].[5: 1]# % LIBA CONTROLLED
    ,SLOT_STAT_CSCA     = STATUS_DATAW[0].[4: 1]# % CSC-A CONTROLL
    ,SLOT_STAT_ACCESS   = STATUS_DATAW[0].[3: 1]# % ACCESS ALLOWED
    ,SLOT_STAT_EXCEPT   = STATUS_DATAW[0].[2: 1]# % ABNORMAL STATE
    ,SLOT_STAT_SIDE2    = STATUS_DATAW[0].[1: 1]# % SIDE 2 LOADED
    ,SLOT_STAT_FULL     = STATUS_DATAW[0].[0: 1]# % TAPE IN SLOT
    ,SLOT_STAT_VOL_TAG  = STATUS_DATAW[1]#        % BAR CODE LABEL
    ,SLOT_STAT_VOL_TAG2 = STATUS_DATAW[2]#        % SN SIDE 2
    ,SLOT_STAT_UNIT     = STATUS_DATAW[3]#        % USING DRIVE
    ;

ARRAY STATUS_DATAW[0] = STATUS_DATA;
```

Function: Returns current status information about a library storage slot.

Usage: This procedure will return the current status information for a library storage slot. The procedure is used to diagnose errors and create information displays for the user.

Parameters:

LIBRARY_ID	(Input) the ID of the library that contains the storage slot.
SLOT	(Input) the storage slot number to return status for.
STATUS_DATA	(Output) the status information is returned as a set of fields. These fields are defined as follows:

Word 0

[47:16]	the element number of the slot as defined by the library hardware
[31:16]	the slot number
[08:01] = 1	if the slot is in Data Domain VTL
[07:01] = 1	if the slot is in back-end library
[06:01] = 1	if the slot is in virtual library
[05:01] = 1	if the slot is controlled by StorageTek Library Attach.

[04:01] = 1	if the slot is controlled by Unisys CSC-A.
[03:01] = 1	if the slot is accessible (can be used).
[02:01] = 1	if the slot has an error reported against it.
[01:01] = 1	if the media has 2 sides and the second side is in use in the drive unit.
[00:01] = 1	if the slot has a cartridge assigned to it.

All other fields in this word are reserved for future use.

Word 1

If the slot has a cartridge assigned to it, this field contains the right most 6 characters of the bar code label of the cartridge. If the library does not have a barcode reader, the serial number of the cartridge is returned if known.

Word 2

If the slot has a cartridge assigned to it, and the media has 2 sides, this is the serial number of the second side otherwise it is zero.

Word 3

If the slot has a cartridge assigned to it and the cartridge is in use by a drive, the drives' unit number is stored here.

Results: Information showing the current status of the referenced drive unit is returned. The drive status is checked.

Possible errors: 15, 16

SLOT_LOAD Procedure

```
BOOLEAN PROCEDURE SLOT_LOAD(LIBRARY_ID, SLOT, DOOR_ID);
VALUE LIBRARY_ID, SLOT, DOOR_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) THAT HAS SLOT
    ,SLOT               % SLOT TO BE LOADED
    ,DOOR_ID            % DOOR TO GET TAPE FROM
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure takes a cartridge from a door and places it in a storage slot.

Usage: This procedure will move a cartridge from a door to a storage slot. It is different from the LIBRARY_MOVE procedure in that it has some additional validity checking and error recovery.

Parameters:	LIBRARY_ID	(Input) the ID of the library having the storage slot to be loaded
	SLOT	(Input) the slot number to put the cartridge into
	DOOR_ID	(Input) the ID of the door to take the cartridge from

Results: The storage slot is loaded with the cartridge from the door. Inventory data is updated.

Possible errors: 10, 11, 16, 19, 26

SLOT_UNLOAD Procedure

```
BOOLEAN PROCEDURE SLOT_UNLOAD(LIBRARY_ID, SLOT, DOOR_ID);
VALUE LIBRARY_ID, SLOT, DOOR_ID;
REAL
     LIBRARY_ID         % LIBRARY ID (HANDLE) THAT HAS SLOT
    ,SLOT               % SLOT TO BE UNLOADED
    ,DOOR_ID            % DOOR TO SEND TAPE TO
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure takes a cartridge from a storage slot and places it in a door.

Usage: This procedure will move a cartridge from a storage slot to a door. It is different from the LIBRARY_MOVE procedure in that it has some additional validity checking and error recovery.

Parameters:	LIBRARY_ID	(Input) the ID of the library that has the storage slot
	SLOT	(Input) the slot number to take the cartridge from
	DOOR_ID	(Input) the ID of the door to send the cartridge to

Results: The door is loaded with the cartridge from the storage slot. Inventory data is updated.

Possible errors: 12, 13, 14, 15, 16, 19, 26

EMPTY_UNIT Procedure

```
BOOLEAN PROCEDURE EMPTY_UNIT(LIBRARY_ID, UNIT_ID);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % ID OF LIBRARY TO SCAN FOR AN EMPTY UNIT
    ,UNIT_ID            % ID OF FIRST EMPTY UNIT FOUND
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure attempts to find an empty and available drive unit in the specified library.

Usage: This procedure finds a drive unit that is empty and available on the MCP executing this procedure. A unit that is FREE, SAVED, or reserved is not selected. If multiple drives are available, the first unit is selected.

Parameters:

LIBRARY_ID	(Input) the ID of the library to search for an empty drive unit
UNIT_ID	(Output) the ID of the drive unit found available

Results: The unit referenced by UNIT_ID is held (not returned by another EMPTY_UNIT call) for 60 seconds. The caller should load the drive unit in that time, or the unit becomes available to other callers. Unit status is checked. If a unit is found in a "limbo" state (drive has cartridge but ejected) the drive is unloaded to a slot.

Possible errors: 7, 15, 26

EMPTY_UNIT_SELECTOR Procedure

```
BOOLEAN PROCEDURE EMPTY_UNIT(LIBRARY_ID, UNIT_ID, MT_SELECTOR);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % ID OF LIBRARY TO SCAN FOR AN EMPTY UNIT
    ,UNIT_ID            % ID OF FIRST EMPTY UNIT FOUND
    ;
REAL PROCEDURE MT_SELECTOR(DRIVES);
  ARRAY DRIVES[0];
  FORMAL;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure attempts to find all empty and available drive units in the specified library. A call-back procedure then selects a specific drive unit.

Usage: This procedure finds all drive units that are empty and available on the MCP system executing this procedure. A unit that is FREE, SAVED, or reserved is not selected. An array of available drives is then passed to the MT_SELECTOR call-back procedure which returns a specific drive from those available.

Parameters:

LIBRARY_ID	(Input) the ID of the library to search for an empty drive unit
UNIT_ID	(Output) the ID of the drive unit found available
MT_SELECTOR	(Call-back) passed an array of available drives, the procedure returns the unit id of a specific drive. The passed array has a count in word 0 followed by count number of unit ids.

Results: The drive id returned by MT_SELECTOR is placed in the UNIT_ID parameter. The unit referenced by UNIT_ID is held (not returned by another EMPTY_UNIT call) for 60 seconds. The caller should load the drive unit in that time, or the unit becomes available to other callers. Unit status is checked. If a unit is found in a "limbo" state (drive has cartridge but ejected) the drive is unloaded to a slot.

Possible errors: 7, 15, 26

EMPTY_SLOT Procedure

```
BOOLEAN PROCEDURE EMPTY_SLOT(LIBRARY_ID, SLOT);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % ID OF LIBRARY TO SCAN FOR AN EMPTY SLOT
    ,SLOT               % NUMBER OF FIRST EMPTY SLOT FOUND
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure attempts to find an empty and available storage slot in the specified library.

Usage: This procedure finds a storage slot that is empty and available.

Parameters:

LIBRARY_ID	(Input) the ID of the library to search for an empty storage slot
SLOT	(Output) the number of the storage slot found available

Results: An available storage slot is returned. Unlike EMPTY_UNIT, a slot is not held (reserved). Therefore, retry logic is suggested when using the returned slot for other procedures. Slot status is checked

Possible errors: 7, 15, 26

EMPTY_DOOR Procedure

```
BOOLEAN PROCEDURE EMPTY_DOOR(LIBRARY_ID, DOOR_ID);
VALUE LIBRARY_ID;
REAL
     LIBRARY_ID         % ID OF LIBRARY TO SCAN FOR AN EMPTY DOOR
    ,DOOR_ID            % ID OF FIRST EMPTY DOOR FOUND
    ;
     LIBRARY LIBRARYSUPPORT;
```

Function:	This procedure attempts to find an empty and available output door in the specified library.	
Usage:	This procedure finds an output door that is empty and available.	
Parameters:	LIBRARY_ID	(Input) the ID of the library to search for an empty storage slot
	DOOR_ID	(Output) the ID of the output door found available
Results:	An available output door ID is returned. Unlike EMPTY_UNIT, a door is not held (reserved). Therefore, retry logic is suggested when using the returned slot for other procedures. Door status is checked	

Possible errors: 7, 15, 26

CONFIG_EVENT_QUEUE Procedure

```
DEFINE
 EVENT_TYPE        = M[0]#
     ,ENTER_EVENTV = 1# % IMPORT OF CARTRIDGE
     ,EJECT_EVENTV = 2# % EXPORT OF CARTRIDGE
     ,LIBRARY_EVENTV=3# % LIBRARY OFFLINE/ONLINE
     ,QUEUE_EVENTV = 4# % QUEUE ATTACHED/DETACHED
     ,MAINT_EVENTV = 5# % LOG MAINT ENTRY
     ,SYNC_EVENTV  = 6# % SYNCHRONIZE WITH LOGPROCESSOR
     ,CART_EVENTV  = 7# % SOMETHING TO KNOW ABOUT A CARTRIDGE
     ,SCHED_EVENTV = 8# % SCHEDULE ACTIVITY FOR LIBRARY STACK
     ,UNIT_EVENTV  = 9# % SOMETHING TO KNOW ABOUT A TAPE UNIT
     ,VTL_EVENTV   =10# % SOMETHING OF INTEREST FROM VTL AGENT
,EVENT_TS          = M[1]# % ALL EVENT QUEUE ENTRIES
,EVENT_SN          = M[2]# % ENTER, EJECT, CART AND UNIT ENTRIES
,EVENT_LIB_STATE   = M[2]# % LIBRARY ENTRIES
     ,ONLINE_EVENTV= 1# % LIBRARY WENT ONLINE
     ,OFFLINE_EVENTV=2# % LIBRARY WENT OFFLINE
     ,INVEN_EVENTV  =3# % LIBRARY INVENTORY CHANGED - MO ONLY
,EVENT_QUEUE_CHANGE= M[2]# % EVENTQ ENTRIES
     ,FIRST_EVENTV = 1# % ATTACH WAS DONE ON QUEUE
     ,LAST_EVENTV  = 2# % OTHER TASK DETACHED FROM QUEUE
,EVENT_UID         = M[2]# % SCHEDULE ENTRIES
,EVENT_CART_TYPE   = M[3]# % CARTRIDGE EVENT ENTRY TYPE
     ,STUCK_EVENTV = 1# % CARTRIDGE WAS FOUND STUCK IN PICKER
,EVENT_LIB_ID      = M[4]# % ENTER, EJECT, SCHEDULE, LIB & ENTRIES
,EVENT_DOOR_ID     = M[5]# % ENTER AND EJECT ENTRIES
,EVENT_MSGSZ       = 6#    % ENTER, EJECT, LIB, QUEUE & UNIT EVENTS
,EVENT_WHO         = M[2]# % MAINT ENTRIES (MIX NUMBER)
,EVENT_WHAT        = M[3]# % MAINT ENTRIES (KIND OF MAINT ENTRY)
,EVENT_WHERE       = M[4]# % MAINT ENTRIES (SEQUENCE NUMBER)
,EVENT_WHEN        = M[5]# % SCHEDULE ENTRIES
,EVENT_MSG         = M[5]# % MAINT ENTRY MSG (TEXT)
,EVENT_UNIT_TYPE   = M[3]# % UNIT EVENT ENTRY TYPE
     ,LOAD_EVENTV  = 1# % UNIT WAS LOADED (NYI)
     ,UNLOAD_EVENTV= 2# % UNIT WAS UNLOADED
,EVENT_UNIT_ID     = M[5]# % UNIT AFFECTED
,EVENT_VTL_TYPE    = M[2]# % VTL EVENT ENTRY TYPE
     ,STACK_DONEV   =1# % VTL STACK PROCESS FINISHED
     ,UNSTACK_DONEV =2# % VTL UNSTACK PROCESS FINISHED
     ,EXPORT_DONEV  =3# % VTL COPY TO TAPE (EXPORT) FINISHED
     ,IMPORT_DONEV  =4# % VTL COPY FROM TAPE (IMPORT) FINISHED
     ,REMCOPY_DONEV =5# % VTL COPY TO VTL (REMOTE COPY) FINISHED
     ,STACK_STARTV  =6# % VTL STACK PROCESS STARTED
     ,UNSTACK_STARTV=7# % VTL UNSTACK PROCESS STARTED
     ,EXPORT_STARTV =8# % VTL COPY TO TAPE (EXPORT) STARTED
     ,IMPORT_STARTV =9# % VTL COPY FROM TAPE (IMPORT) STARTED
     ,REMCOPY_STARTV=10#% VTL COPY TO VTL (REMOTE COPY) STARTED
     ,TAPE_CREATEV  =11#% VTL VIRTUAL TAPE CREATED
     ,TAPE_DESTROYV =12#% VTL VIRTUAL TAPE DELETED
     ,STACK_FAILEDV =13#% VTL STACK PROCESS FAILED
     ,UNSTACK_FAILEDV=14#%VTL UNSTACK PROCESS FAILED
     ,EXPORT_FAILEDV=15#% VTL COPY TO TAPE (EXPORT) FAILED
     ,IMPORT_FAILEDV=16#% VTL COPY FROM TAPE (IMPORT) FAILED
     ,REMCOPY_FAILEDV=17#%VTL COPY TO VTL (REMOTE COPY) FAILED
,EVENT_VTL_LIBID   = M[3]# % VTL EVENT LIBRARY_ID
,EVENT_VTL_BESN    = M[4]# % VTL EVENT BACK END TAPE SN
,EVENT_VTL_VTLSN   = M[5]# % VTL EVENT VLT TAPE SN
,EVENT_VTL_OLDBC   = M[5]# % VTL EVENT -REDEFINE- FOR UNSTACK
,EVENT_VTL_SLOT    = M[6]# % VTL EVENT SLOT WHERE TAPE PLACED
,EVENT_VTL_NEWBC   = M[6]# % VTL EVENT -REDEFINE- FOR UNSTACK
,EVENT_VTL_JOBNUM  = M[7]# % VTL EVENT VTL JOB NUMBER
```

```
,EVENT_MAINT_HDRSZ = 5#% MAINT ENTRY HEADER SIZE IN WORDS
,EVENT_MAINT_MSGSZ =(50+EVENT_MAINT_HDRSZ)#% MAX MAINT ENTRY MSG SZ

% EVENT QUEUE OPTIONS
,UNSOLICITED_IMPORTSF  = [1:1]#
,ALL_IMPORTSF          = [2:1]#
,UNSOLICITED_EXPORTSF  = [3:1]#
,ALL_EXPORTSF          = [4:1]#
,LIBRARY_STATEF        = [5:1]#
,INVEN_ON_ATTACHF      = [6:1]#
,INVEN_ON_ENABLEF      = [7:1]#
,ATTACH_DETACHF        = [8:1]#
,LOGMAINTF             = [47:1]# % ALWAYS SET

% CONFIG_EVENT_QUEUE WHAT OPTIONS
,GET_OPTIONSV = 0#
,SET_OPTIONSV = 1#
,DETACHV      = 2#
;

BOOLEAN PROCEDURE CONFIG_EVENT_QUEUE(OPTIONS,WHAT);
VALUE WHAT;
REAL OPTIONS, WHAT;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure sets or interrogates the options the caller wishes to see in the event queue defined in SET_EVENT_QUEUE. It can also detach the caller's queue from the event queue.

Usage: Note: Events 5, 9, and 10 are always returned and cannot be turned off. Most events 1-4 and 6-9 are not needed if the calling media manager does not keep local state of library information (recommended) but rather calls the LibraryManager API as needed to get current state.

Parameters:

OPTIONS	(Input/Output) the sets or returns the type of event messages to be placed in the event queue
WHAT	(Input) the action to take. Zero returns the options currently set. One sets the options in the OPTIONS word. Two causes the callers queue to be detached from the event queue.

Results: Event queue options are set or returned or the caller's queue is detached from the event queue.

Possible errors: TRUE if the WHAT parameter is invalid.

NOTE: The EVENT_TYPE VTL_EVENTV with VTL_EVENT_TYPE of REMCOPY_STARTV or REMCOPY_DONEV attempt to return the bar code of the tape being remote Copied. If the VTLSUPPORT software is restarted or there is an error requesting information from the VTL Agent, then this field will be empty.

SET_EVENT_QUEUE Procedure

```
BOOLEAN PROCEDURE SET_EVENT_QUEUE(CALLERQ);
QUEUE CALLERQ;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure provides a mechanism to see unsolicited inventory changes in a library.

Usage: This procedure attaches the CALLERQ to a queue in the TAPELIBRARY support library. Event messages are placed in this queue for handling by the caller. Only one user at a time may attach to this queue. This procedure should only be used if the caller is keeping its own inventory state data, otherwise rely on the above procedure calls to ensure accurate real-time status.

Parameters: CALLERQ (Input/Output) the queue to place event messages in

Results: The caller's queue and an internal queue are attached such that they become a single queue. The TAPELIBRARY support library will place event messages in this queue to notify the caller of certain events. See defines described in CONFIG _EVENT_QUEUE above for the layout of the event messages.

Possible errors: TRUE if another caller already attached to this queue.

DEBUG_INFO Procedure

```
PROCEDURE DEBUG_INFO(PINFO);
VALUE PINFO;
POINTER PINFO;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure sets the trace file naming parameters for the DEBUG_TRACE function.

Usage: This procedure must be called once before DEBUG_TRACE is called.

Parameters: PINFO (Input) points to a string with the following format

DEBUG.<debug serial number>.<debug family>.<debug usercode>. (null)
Example: DEBUG.014290.DSIPACK.MTM.(null)

Results: The above example will cause trace files to be created as follows.

*<debug usercode>/DEBUG/TRACE/<debug serial number>/<library name>/<date>/<time> ON <debug family>

*MTM/DEBUG/TRACE/014290/VTLSUPPORT/20101209/072245 ON DSIPACK

*MTM/DEBUG/TRACE/014290/TAPELIBRARY/20101209/072245 ON DSIPACK

*MTM/DEBUG/TRACE/014290/TAPEMANAGER/20101209/072245 ON DSIPACK

*MTM/DEBUG/TRACE/014290/"DSI CONTROLLER"/20101209/072245 ON DSIPACK

Notes: 1) Any dumps taken will have a title of *MTM/DEBUG/PDUMP/014290/=

2) Notice that the usercode is not (MTM) but *MTM due to an MCP bug when taking dumps under a usercode

3) The <debug serial number> must be 6 digits

4) Once you have set the usercode and family you can shorten the string passed to DEBUG_INFO to simply DEBUG.014291.(null) to change the debug serial number

Possible errors: None

DEBUG_TRACE Procedure

```
PROCEDURE DEBUG_TRACE(ON_OR_OFF);
VALUE ON_OR_OFF;
BOOLEAN ON_OR_OFF;
     LIBRARY LIBRARYSUPPORT;
```

Function: This procedure activates or deactivates the reporting of debug tracing information when procedures in the TAPELIBRARY support library are called.

Usage: This procedure is used to diagnose possible problems in the TAPELIBRARY support library.

Parameters:

ON_OR_OFF	(Input) TRUE to active debug tracing FALSE to deactivate debug tracing

Results: When debug tracing is activated, a file is created that contains information about the activity of the TAPELIBRARY support library. If the library is attached to the DSICONTROLLER support library, its debug trace routines are also activated. When tracing is deactivated, this file may be printed by TapeManager, CANDE, or any program capable of printing simple text files.

Possible errors: None

Glossary

A

attribute

1. A quality added to text to make it stand out from surrounding text, such as underlining, boldface, subscript, superscript, or struck out text. Attributes can be combined so that a character or text has several attributes at the same time.
2. A configurable quality used to define a file or station and so on.

C

command

an instruction to a computer to perform a special task.

COMS

Unisys A-Series Communications Management System.

cursor

The blinking underline or block on the screen that indicates where the next character can be entered. As characters are entered, the cursor moves to the right.

D

DELETE

Display data control key which enables you to delete the character at the current cursor position. Characters to the right of the cursor within the same field and on the same line shift one character position to the left.

F

field

1. An area on a screen or form in which data is displayed or entered. The delimiters of the field can be visible or invisible to the terminal operator.
2. A consecutive group of bits within a word or a component of a record that represents a logical piece of data.

File

A named group of related records.

form

A special screen containing prompts requesting information and empty form fields in which the requested information can be entered.

H

HOME

A field cursor movement key that moves the cursor to the home position, the first unprotected character position of the screen. If the display has no unprotected fields or no fields at all, the home position is row 1, column 1.

I

Input

Text typed into the computer.

M

Magneto-Optical
A type of storage media and drive that uses a combination of laser and magnetic fields to store and retrieve data.

MCP
The Master Control Program or operating system of a Unisys A-Series system.

menu
A special group of fields that show the user a set of options from which to choose.

message boxes
Message boxes are popup windows that provide status and error messages.

message line
Displays the INSERT operator guidance message

mix number
A number assigned to a job or task by the A-Series operating system (MCP).

MO
See Magneto-Optical.

P

pack
A disk drive on a Unisys A-Series system.

parameter
A data item provided to or by a program or procedure (subroutine).

Process
1. Execution of a program or procedure.
2. A software application; activity or series of operations that produces specified results.

R

railroad diagram
A graphic representation of the syntax of a command or statement.

S

screen
An image that appears on the display area of a terminal or workstation prompting the user to enter data, displays information or presenting options from which to choose.

scroll
To move forward and backward within a list, within help text, or within other displayed items.

session
A session is a Logical connection between two units on a network.

slot
A generic term for a cartridge storage location within an automated cartridge library.

Specify
A key on a Unisys T-27 (or compatible) terminal that, when pressed, sends a message containing the current cursor position to the host system.

station
: The outer end of a communication line. A station can correspond to a single terminal connected on a single line, or several stations can be connected on a line.

system
: Operating system

T

tape library
: A hardware device containing multiple magnetic tape cartridges along with robotics to load them into one or more tape drives.

terminal
: I/O device designed to receive or send information in a network.

U

user
: Individual accessing the computer.

User code
: The <string> used to identify the user to the host system.

user interface
: The appearance of a program to a user.

Index

Document Evaluation Form

DSI is interested in receiving your comments and suggestions regarding this document. Comments will be utilized in subsequent revisions to improve this document.

Manual Title:

Cartridge Library Installation Guide
for
Unisys MCP Systems

Version No: **Release 10.070.0641**
Date: **June 2023**

Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments / Suggestion:

____
____
____
____
____
____
____
____
____
____
____
____

From:

Name: ____

Title: ____

Company:____

Address: ____

City/ST/ZIP:____

Phone: (_____)_____-_________ Date: _____/_____/_____

Remove form and mail preaddressed overleaf, or FAX to:

Dynamic Solutions International (DSI)
Product Development Group

FAX Number: (303)754-2066

FOLD HERE AND TAPE

PUT
STAMP
HERE

Dynamic Solutions International (DSI)
8744 Lucent Blvd. Suite 106
Highlands Ranch, Colorado 80129
U.S.A.

Attn: Product Development Group