StorageTek®

PN 312537401

OS 2200 Client System Component User Interface (CSCUI)

5R1 Programmer's Reference Manual

February 2005

THIS DOCUMENT DOES NOT EXTEND OR CREATE WARRANTIES OF ANY NATURE, EITHER EXPRESSED OR IMPLIED. Storage Technology Corporation cannot accept any responsibility for use of the information in this document or of any associated software program. Storage Technology Corporation assumes no responsibility for any data corruption or erasure as a result of the information in this document, including any associated software program. You are responsible for backing up your data. You should be careful to ensure that use of the information complies with all applicable laws, rules, and regulations of the jurisdictions with respect to which it is used.

Information in this document is subject to change, and Storage Technology Corporation reserves the right to revise or modify the information in this document without prior notification. In the event of changes, this document will be revised. Contact Storage Technology Corporation to verify that you have the most current version of this document.

Comments concerning the information in this document should be directed to:

Storage Technology Corporation
2270 South 88th Street
Louisville, CO 80028-4232
(800) 678-4430

CONTENTS

PREFACE

Purpose xi
Audience xi
How to Use This Document xi
Command Syntax Notation xii
Related Documentation xiii

1. INTRODUCTION

Overview 1-2

2. CSC - TLMS INTERFACE

TLMS Functions 2-2
Server Functions 2-2
TLMS and Server Interactions 2-3
    Cartridge Tape Inventory 2-3
    Cartridge Tape Ownership 2-3
    Scratch Cartridge Tapes 2-3
Cartridge Tape Movement 2-6
    Entering Cartridge Tapes Into the ACS 2-6
    Ejecting Cartridge Tapes From the ACS 2-6
    Multiple Eject Requests 2-8
    Suggestions for Multiple Concurrent Ejects 2-9

3. CSCUI CONTROL FUNCTIONS

Calling CSCUI Control Functions 3-2
    MASM Calling Interface 3-2
    UCS C and UCOB Calling Interface 3-3
CSCUI Control Function Restrictions 3-4

CSCUI Services....3-4
VALID-FIELDS-PASSED Field....3-4
WAIT-TIME and VALID-FIELDS-RETURNED Fields....3-4
Reloading the Common Bank....3-4
Addressing CSCUI Control Functions....3-5
Using CSCUI....3-5
CSCUI Packet Formats....3-6
Control Function Descriptions....3-6
REGISTER Control Function....3-6
DEREGISTER Control Function....3-9
TERMINATE-REQUEST Control Function....3-11

4. CSCUI USER REQUESTS

User Request Protocol....4-2
DO-ENTER USER REQUEST....4-2
DO-VOLRPT USER REQUEST....4-6
Volume Report Parameters....4-6
Request Packet....4-14
INITIATE-EJECT User Request....4-17
Using INITIATE-EJECT User Requests....4-17
MOUNT User Request....4-23
SCRATCH/UNSCRATCH User Request....4-30
VOLUME-INFORMATION User Request....4-35

5. CSCUI NOTIFICATION EXITS

Notification Exit Protocol....5-2
AFTER-DISMOUNT Notification Exit....5-2
AFTER-EJECT Notification Exit....5-7
AFTER-ENTER Notification Exit....5-11
AFTER-MOUNT Notification Exit....5-14
BEFORE-MOUNT Notification Exit....5-19
EJECT-COMPLETION Notification Exit....5-23

6. CSCUI LOGGING

Overview....6-2
CSCUI Logging Functions....6-2
CSCUI Log Record Format....6-3
What log records are generated....6-5

APPENDIX A. REFERENCE TABLES

VALID-FIELDS-PASSED Field Values....A-2

CSCUI-STATUS Codes A-3
CSCUI Exit IDs A-3

APPENDIX B. SAMPLE PROGRAMS

CSCUI packet definitions B-2
CSCUI Interface Functions B-2
    ACOBCSCUI B-2
    UCS$CSCUI B-3
    UCS C B-3
    UCS COBOL B-4
    LINKing and MAPping B-4
Sample Programs B-5

INDEX

EFFECTIVE PAGES

TABLES

Table 2-1. ACS and TLMS Scratch Status Interactions 2-4

Table 3-1. Registers Used by CSCUI Control Functions 3-2
Table 3-2. REGISTER Control Function Field Descriptions 3-7
Table 3-3. DEREGISTER Control Function Field Descriptions......................3-10
Table 3-4. TERMINATE-REQUEST Control Function Field Descriptions....3-12

Table 4-1. DO-ENTER User Request Field Descriptions 4-4
Table 4-2. Volume Report Parameters..4-7
Table 4-3. FORMAT Statement Fields...4-9
Table 4-4. DO-VOLRPT User Request Field Descriptions.............................4-15
Table 4-5. INITIATE-EJECT User Request Field Descriptions 4-20
Table 4-6. Mount Types and Request Packet Field Requirements...................4-24
Table 4-7. MOUNT User Request Field Descriptions......................................4-26
Table 4-8. Scratch Pool Indicator Value/ID Reference (ACSLS only) 4-30
Table 4-9. SCRATCH/UNSCRATCH User Request Field Descriptions.........4-32
Table 4-10. VOLUME-INFORMATION User Request Field Descriptions....4-36

Table 5-1. AFTER-DISMOUNT Notification Exit Field Descriptions..............5-4
Table 5-2. AFTER-EJECT Notification Exit Field Descriptions 5-8
Table 5-3. AFTER-ENTER Notification Exit Field Descriptions....................5-12
Table 5-4. AFTER-MOUNT Notification Exit Field Descriptions 5-16
Table 5-5. BEFORE-MOUNT Notification Exit Field Descriptions 5-20
Table 5-6. EJECT-COMPLETION Notification Exit Field Descriptions 5-24

Table 6-1. CSCUI Log Record Subtypes..6-3
Table 6-2. CSCUI Log Record Field Descriptions...6-4

Table A-1. VALID-FIELDS-PASSED Field Definitions..................................A-2
Table A-2. CSCUI-STATUS Codes..A-3
Table A-3. CSCUI Exit IDs..A-4

Table B-1. Sample Programs Provided with CSC B-6

FIGURES

Figure 1-1. CSC User Interface Overview 1-2

Figure 3-1. REGISTER Control Function Packet 3-6
Figure 3-2. DEREGISTER Control Function Packet 3-9
Figure 3-3. TERMINATE-REQUEST Control Function Packet 3-11

Figure 4-1. DO-ENTER User Request Packet 4-3
Figure 4-2. DO-VOLRPT User Request Packet 4-14
Figure 4-3. INITIATE-EJECT User Request Packet 4-19
Figure 4-4. MOUNT User Request Packet 4-25
Figure 4-5. SCRATCH/UNSCRATCH User Request Packet 4-31
Figure 4-6. VOLUME-INFORMATION User Request Packet 4-35

Figure 5-1. AFTER-DISMOUNT Notification Exit Packet 5-3
Figure 5-2. AFTER-EJECT Notification Exit Packet 5-7
Figure 5-3. AFTER-ENTER Notification Exit Packet 5-11
Figure 5-4. AFTER-MOUNT Notification Exit Packet 5-15
Figure 5-5. BEFORE-MOUNT Notification Exit Packet 5-19
Figure 5-6. EJECT-COMPLETION Notification Exit Packet 5-23

Figure 6-1. CSCUI Log Record Format 6-4

PREFACE

PURPOSE

This is the *OS 2200 Client System Component User Interface (CSCUI) Programmer's Reference Manual*. This manual describes the user interface component of CSC. CSC is the software used by the Unisys 1100/2200 Client System (the "client") to communicate with the StorageTek® Library Control System (the "server") and the Automated Cartridge System (ACS). CSCUI facilitates the exchange of information between user programs and the server.

AUDIENCE

This guide is written primarily for *programmers*. It assumes that you are familiar with the following hardware and software components:

- Client System Component (CSC)
- ACS
- Unisys Series 1100 and 2200 computers
- Executive Control Language (ECL)
- Solaris®-based Library Control System or Nearline Control System

HOW TO USE THIS DOCUMENT

Chapter 1. Introduction

This chapter provides a brief introduction and overview of CSCUI components, operations, and terminology.

Chapter 2. CSC - TLMS Interface

This chapter explains how CSCUI interacts with a Tape Library Management System (TLMS). Included are discussions of special issues you should consider when implementing a TLMS - CSC interface.

Chapter 3. CSCUI Control Functions

This chapter discusses CSCUI Control Functions (e.g. REGISTER, DEREGISTER, and TERMINATE-REQUEST) required to register operations with CSCUI. Each explanation includes a description of the function, a packet diagram, and a table detailing fields passed and returned.

Chapter 4. CSCUI User Requests

This chapter describes the CSCUI User Requests (e.g., DO-ENTER, DO-VOLRPT, INITIATE-EJECT, MOUNT, SCRATCH/UNSCRATCH, and VOLUME INFORMATION). Each explanation includes a description of the user request, a packet diagram, and a table detailing fields passed and returned.

Chapter 5. CSCUI Notification Exits

This chapter describes the CSCUI Notification Exits (e.g., AFTER-DISMOUNT, AFTER-EJECT, AFTER-ENTER, AFTER-MOUNT, BEFORE-MOUNT, and EJECT-COMPLETION). Each explanation includes a description of the notification exit, a packet diagram, and a table detailing fields passed and returned.

Chapter 6. CSCUI Logging

This chapter describes CSCUI logs and logging functions.

Back Matter

This manual also includes an appendix of reference tables, an appendix of sample programs that use CSCUI, and an index.

COMMAND SYNTAX NOTATION

This manual uses the following conventions for representing command syntax notation and message displays:

UPPERCASE	indicates a command or keyword.

lowercase italic indicates a user- or system-supplied variable value. For example, in XX=*userid*, you enter the actual userid for *userid*.

abbreviation indicates a command that can be abbreviated to its minimum acceptable form. For example, ENAble can be abbreviated to ENA.

vertical bar | separates operand alternatives. For example, A | B indicates that you must select either A or B.

brackets [] indicate an option that can be omitted. For example, [A | B | C] indicates that you can select A, B, C, or nothing.

braces {} indicate an option that you *must* choose. For example, {A | B} indicates that you must choose either A or B.

underlining indicates the system default. If you do not enter a parameter or value, the system will supply the underscored value. For example, A | B | C indicates that if you do not choose an option, the system will default to C.

ellipses ... indicate that entries can be repeated as often as necessary.

SMALLCAPS indicate a key, such as XMIT or F1.

RELATED DOCUMENTATION

OS 2200 Client System Component (CSC) Technical Bulletin, Storage Technology Corporation (312537701)

OS 2200 Client System Component (CSC) Installation Guide, Storage Technology Corporation (313471401)

OS 2200 Client System Component (CSC) System Administrator’s Guide, Storage Technology Corporation (312537501)

OS 2200 Client System Component (CSC) Operations Guide, Storage Technology Corporation (312537201)

OS 2200 Client System Component (CSC) Client Direct Interconnect (CDI) Troubleshooting Guide, Storage Technology Corporation (312537601)

OS 2200 Client System Component (CSC) User Reference Manual, Storage Technology Corporation (312537801)

OS 2200 Client System Component (CSC) UNISYS OS 2200 CSC 5R1 VSM Reference, Storage Technology Corporation (312537901)

Automated Cartridge System Library Software Product Document Set for Solaris, Storage Technology Corporation.

Nearline Control Solution 4.0 Publication Kit, Storage Technology Corporation (313456301)

Exec System Software Operations Reference Manual, Unisys Corporation (7831 0281)

Executive Control Language (ECL) and FURPUR Reference Manual, Unisys Corporation (7830 7949)

Exec System Software Executive Requests Programming Reference Manual, Unisys Corporation (7830 7899)

COMUS End User Reference Manual, Unisys Corporation (7830 7758)

Software Library Administrator (SOLAR) End User Reference Manual, Unisys Corporation (7831 0604)

1. INTRODUCTION

The Client System Component User Interface (CSCUI) is a feature of the Client System Component (CSC). CSCUI implements a general interface between either the Nearline Control Solution or the Solaris-based Library Control System (the “server”) and a Tape Library Management System (TLMS).

CSCUI runs on the Unisys 1100/2200 Client System (the “client”). CSCUI lets user programs exchange information with the server and monitor operations of the Automated Cartridge System (ACS).

CSCUI passes data, user requests, and notification exits through the CSC Task Manager (CSCTM). CSCUI does this by controlling and synchronizing requests from multiple programs. CSCUI does not add noticeable overhead to CSC operations on the client.

OVERVIEW

Figure 1-1 shows a high-level overview of CSCUI and the components with which it interacts.

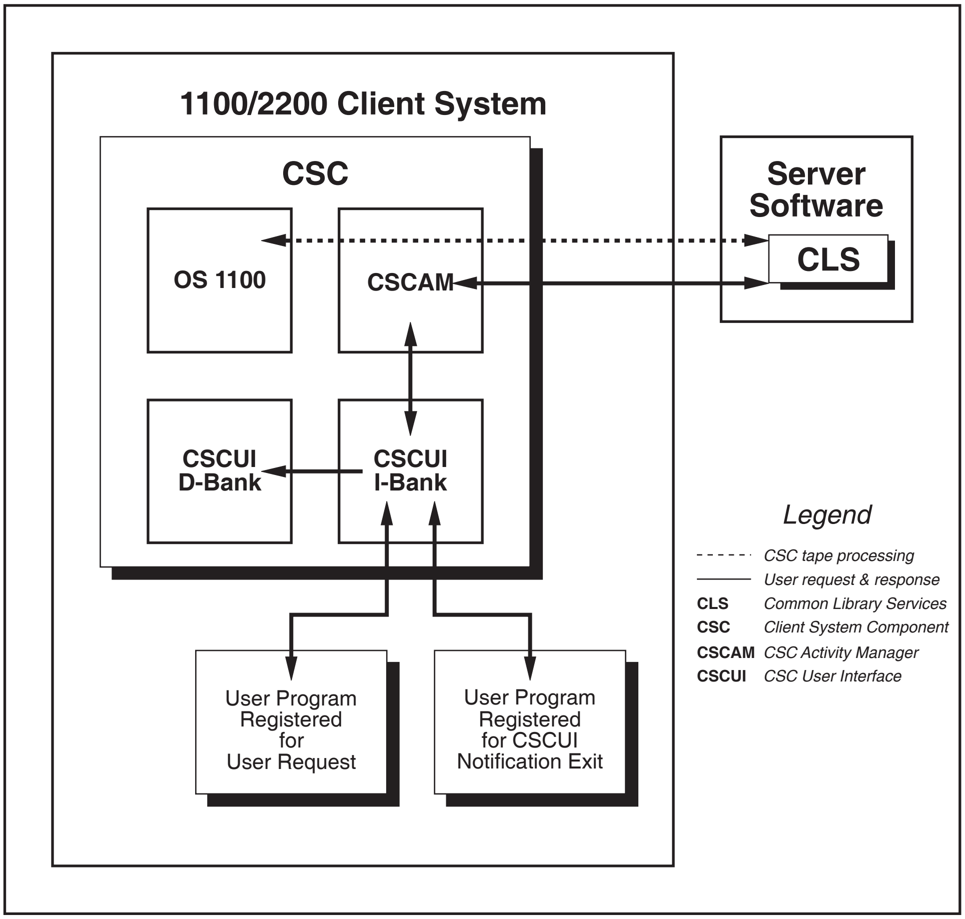
1100/2200 Client System
CSC
OS 1100
CSCAM
CSCUI D-Bank
CSCUI I-Bank
Server Software
CLS
User Program Registered for User Request
User Program Registered for CSCUI Notification Exit
Legend
- - - - - CSC tape processing
——— User request & response
CLS Common Library Services
CSC Client System Component
CSCAM CSC Activity Manager
CSCUI CSC User Interface

Figure 1-1. CSC User Interface Overview

CSCUI lets user programs access CSC and the server to:

- Send requests for server services, such as scratching or unscratching a cartridge tape.

- Receive messages of significant cartridge tape processing events in the ACS, such as cartridge tape mounts and dismounts.

CSCUI uses control and data transfer functions to implement CSC notification exits and processes user requests. A request packet defines the function to be performed. All information required to complete the function, and all returned data, are passed in that packet.

2. CSC - TLMS INTERFACE

This chapter presents the functional requirements of a TLMS interface to the server. It describes the following:

- General TLMS and server operations
- Interaction between a TLMS and the server
- CSCUI facilities that execute the required functions of a TLMS

TLMS FUNCTIONS

A TLMS controls cartridge tape use at a site. It can be automated, manual, or a combination of both. Regardless of the approach used, a TLMS:

- Keeps an inventory of all cartridge tapes
- Protects cartridge tape ownership
- Maintains one or more pools of available scratch cartridge tapes
- Produces reports showing various aspects of cartridge tape use

A TLMS may also direct or assist in the following:

- Transfer of cartridge tapes to and from off-site storage location(s)
- Scheduling cartridge tapes for cleaning and/or replacement

SERVER FUNCTIONS

The ACS stores and provides physical handling of cartridge tapes under the direction of software on the server. Cartridge tapes move within a Library Storage Module (LSM) in the ACS and can occupy one of the following places:

- a cartridge drive
- a storage cell within the LSM
- a pass-through port to another LSM
- a Cartridge Access Port (CAP)

The ACS Library Software (ACSLS or NCS) controls cartridge tape movement within the ACS. ACSLS or NCS, executing on the server, maintains information about every cartridge tape in the ACS, including location and scratch status. Upon request, ACSLS or NCS performs the following services:

- Mounts a specific cartridge tape on a cartridge drive
- Mounts a scratch cartridge tape on a cartridge drive
- Dismounts a cartridge tape from a cartridge drive
- ENTERs a cartridge tape through the CAP into the ACS
- EJECTs a cartridge tape from the ACS through the CAP
- Changes the ACSLS or NCS scratch status of a cartridge tape

During the course of normal operations, a cartridge tape access request may cause changes in that cartridge tape’s storage location and scratch status. Client-initiated requests may also change the scratch status of a cartridge tape.

TLMS AND SERVER INTERACTIONS

Cartridge Tape Inventory

A TLMS tracks all cartridge tapes at a site. By contrast, ACSLS or NCS maintains information only on cartridge tapes physically within the ACS. There is no direct interaction between these inventories.

The TLMS should have the authoritative status on the location of cartridge tapes which it tracks. The TLMS can use facilities in CSCUI to track cartridge tape movement in and out of the ACS. By carefully using the AFTER-ENTER and AFTER-EJECT notification exits, the TLMS can monitor the movement of cartridge tapes between the ACS and other locations. When you enter a cartridge tape into the ACS, a TLMS registered with the AFTER-ENTER exit will receive notification of the ENTER. The ACS-ID and LSM-ID are included in the data returned to the user within the AFTER-ENTER exit. The TLMS can update its inventory information to reflect the new ACS location. When the cartridge tape is EJECTed, the TLMS is informed via the AFTER-EJECT notification. The TLMS can then update its inventory information.

Cartridge Tape Ownership

ACSLS or NCS does not support cartridge tape ownership. All cartridge tapes in an ACS are physically accessible by any requester. The TLMS is solely responsible for cartridge tape ownership concerns.

Scratch Cartridge Tapes

Scratch cartridge tapes are those on which any requester may write data. Any cartridge tape containing useful data should not be “scratch.” The tape creator is responsible for protecting data by following rules that govern scratch status.

Client View of Cartridge Tape Scratch Status

OS 2200 considers cartridge tapes to be scratch tapes if:

- the expiration date of a labeled cartridge tape has passed.
- a cartridge tape is unlabeled.

A TLMS defines a cartridge tape as “in use” based on retention information provided when the tape is created. A cartridge tape with a zero days retention period is always considered scratch, even if it’s been written to. An in-use cartridge tape reverts to scratch status when its retention period expires.

ACS View of Scratch Status

ACSLS or NCS maintains a scratch status for each cartridge tape within the ACS. It also associates each cartridge tape with a numbered scratch pool. Used together, these pieces of information determine which cartridge tapes will be used to satisfy a MOUNT SCRATCH request from a client. However, the fact that a cartridge tape is not a scratch tape from the ACSLS or NCS's point of view does *not* imply data protection.

ACSLS or NCS cannot access the retention information provided by the cartridge tape creator, nor can it read the data written to the cartridge tape label. As a result, ACSLS or NCS cannot maintain cartridge tape status information like the client does.

ACSLS or NCS changes the status of a cartridge tape as "in use" when it is mounted for any reason. When a cartridge tape is entered into the ACS, it is also assigned an "in-use" status. A cartridge tape can return to scratch status only through a request from the ACSLS Command Processor, or through a program interface from the client. The scratch pool association can be updated as part of the scratch status change.

Table 2-1 shows the interactions of scratch status changes on the TLMS and ACS.

Table 2-1. ACS and TLMS Scratch Status Interactions

		TLMS Cartridge Tape Scratch Status			
		Scratch (no change)	**Scratch to non-scratch**	**Non-scratch (no change)**	**Non-scratch to scratch**
ACS Cartridge Tape Scratch Status	**Scratch (no change)**	no change	out of synch[1]	out of synch[1]	synch reestablished
	Scratch to non-scratch	out of synch[2]	synchronized status change	synch reestablished	out of synch before and after changes[3]
	Non-scratch (no change)	out of synch[1]	synch reestablished	no change	out of synch[1]
	Non-scratch to scratch	synch reestablished	out of synch before and after changes[3]	out of synch[1]	synchronized status change

[1] To guarantee protection of data, defer to the TLMS scratch status. To restore synchronization, either scratch or unscratch the cartridge tape in the ACS.

[2] If the change to non-scratch status on the ACS is the result of an ENTER request, then scratch the cartridge tape in the ACS. The only other way of effecting this change is manually. In this case, the originator of the change must determine which status should be updated.

[3] Both scratch status changes cannot be caused by a single event. This case should be treated like two consecutive scratch status changes and handled accordingly.

Maintaining Scratch Synchronization

The user TLMS should be considered the final authority on the scratch status of a cartridge tape. There are several functions that you can add to the TLMS to help manage the scratch pool. When the TLMS scratches or unscratches a cartridge tape, you should send a SCRATCH-REQUEST or UNSCRATCH-REQUEST to the library server software via CSCUI. This is referred to as an “on-line” method of maintaining scratch synchronization.

When a cartridge tape has been written, its disposition according to the ACS will always be “in-use,” regardless of the retention period specified by the user. To ensure that cartridge tapes are returned to the scratch pool in a timely manner, you can program your TLMS to evaluate a cartridge tape after it is used, and reset its TLMS scratch status. For example, based on criteria such as the application, the account id, and the retention period (within the TLMS), the TLMS may decide that the cartridge tape should remain scratch after it is used. The AFTER-DISMOUNT notification provides additional information to the TLMS about the cartridge tape and the job using the cartridge tape.

Scratch Pools

Scratch pools are groupings of cartridge tapes controlled by the library server software. CSC requires two scratch pools, one for labeled cartridge tapes and another for unlabeled cartridge tapes. Additional scratch pools can be defined. The site administrator configures the scratch pools on the server, and sets the scratch pool ID for each cartridge tape.

In ECL, the scratch pool is referenced by the presence or absence of the “J” option in combination with the CTL-POOLNAME field on the @ASG ECL statement. In CSCUI requests, it is referenced by the scratch-pool-indicator, extended-scratch-pool, and CTL-POOLNAME fields in the CSCUI scratch request packet. The CSC configuration contains information to map the default and CTL-POOLNAMEs with the server scratch pool. The site administrator establishes these mappings. Please refer to the *CSC User Reference Manual Guide* for additional information about using CTL-POOLNAMEs on @ASG ECL statements, and to the *CSC Installation Guide* for information on the CSC configuration parameters.

CARTRIDGE TAPE MOVEMENT

The server moves cartridge tapes into or out of the ACS. CSCUI provides a user interface to enter or eject tapes. Also, CSCUI provides notification of these movements.

Entering Cartridge Tapes Into the ACS

Cartridge tapes enter an ACS through a Cartridge Access Port (CAP). This requires several distinct manual operations. First, the operator must issue commands on the system or server consoles to place the CAP into ENTER mode. Next, the operator must select the cartridge tapes from some storage location and load the CAP with batches of cartridge tapes. When the CAP is closed, the ACS robotic arm moves the cartridge tapes from the CAP to storage locations inside the ACS.

After all of the cartridge tapes have been entered into the ACS, there are several potential interactions with the TLMS. When the cartridge tape is entered, the server software may create a new cartridge tape record and mark the cartridge tape as “in-use.” However, the TLMS may consider it a scratch cartridge tape. By using the AFTER-ENTER notification exit, the TLMS can make decisions about which cartridge tapes to restore to scratch status. The TLMS can use the SCRATCH-REQUEST function in CSCUI to set the scratch status in the ACS.

NOTE

There are no AFTER-ENTER notifications for cartridges entered via the server console or from another client system.

The Enter Command

You can manually enter commands to enter cartridge tapes into the ACS. Alternatively, a user program can submit a DO-ENTER user request via CSCUI.

Ejecting Cartridge Tapes From the ACS

Ejecting cartridge tapes from the ACS requires several distinct actions. First, you must determine which cartridge tapes to eject, then enter commands at the system or server console to eject the tapes from the ACS. Finally, you must coordinate with an operator at the site to physically remove the cartridge tapes from the CAP.

Optional Ways to Eject Cartridge Tapes

You can use one of three ways to eject cartridge tapes:

- manual
- batch
- on-line

These methods differ in how the cartridges are selected and how the selection is passed to the library server.

Manual Ejects

Using a manual approach, you create a list of cartridge tapes to eject manually. An operator then uses the system or server console to eject the cartridge tapes you've specified.

Batch Ejects

In a batch method, user applications utilize standard Executive Request (ER) functions to obtain volume serial numbers ("volsers"). For example, using the file name as a search key, an application can obtain the volser from the Master File Directory (MFD). For a multiple cartridge tape file, this will return all volsers.

An application can also obtain volsers from the File Administration System (FAS). For example, a user might want to select all cartridge tapes from a specific backup generation. By searching on a backup number, an application can obtain the volsers of all cartridge tapes associated with that backup.

On-line Ejects

In the on-line approach, CSCUI allows you to eject cartridge tapes immediately after they are used. The AFTER-DISMOUNT notification exit informs the TLMS when a cartridge tape has been dismounted, and returns to the user program the file qualifier, file name, run-id, account number, and user id. You can implement decision logic in the TLMS to monitor the AFTER-DISMOUNT notification exit and to compare the returned data with predefined selection criteria. If the AFTER-DISMOUNT notification exit data satisfies the selection criteria, the TLMS can use the volser of the cartridge tape to build a CSCUI Eject request packet. The TLMS can then send this request via CSCUI to eject the cartridge tape(s).

The Eject Command

You can manually enter commands to eject individual cartridge tapes or ranges of cartridge tapes from the ACS. The server may perform several eject operations, depending on the capacity of the selected CAP.

Alternately, a user program (such as a TLMS) can submit an Eject request via CSCUI. CSCUI restricts the number of cartridge tapes that can be ejected in one request. CSCUI does not support ranges of cartridge tapes in the Eject request.

Physically Removing Cartridge Tapes

When an eject action completes and all of the cartridge tapes have been placed in the CAP, an operator must physically remove the cartridge tapes from the CAP. Coordination is crucial when you’ve used program control, via CSCUI, to perform the ejects.

After the cartridge tapes have been removed from the CAP, CSCUI provides the EJECT-COMPLETION notification for each ejected cartridge. When ejects are done under program control, the TLMS can match these notifications with the original INITIATE-EJECT user request packet and identify any anomalies, such as cartridge tapes that were not found.

NOTE

*There are no AFTER-EJECT notifications for cartridge tapes ejected via the server console or from another client system, or from the *CSC EJECT keyin.*

Multiple Eject Requests

Because CSCUI permits more than one user to register to the INITIATE-EJECT user request, it is possible for users to issue more than one INITIATE-EJECT concurrently.

NOTE

While multiple users can register for the INITIATE-EJECT user request, only one user can register for the AFTER-EJECT notification exit. That means only one user program can receive these notifications.

- If a CAP was not specified on the eject command, then the library server software will select the available CAP with the highest priority, and use it for the duration of that eject. The server will not select a CAP with a priority of zero.

- Site specific settings determine what action is taken in the following situation: The specified CAP is not available, or the number of ejects submitted is greater than the number of available CAPs. The possible settings are the eject request is queued in CSC until the CAP is available, or the request terminates with a request completion status of 9.

Suggestions for Multiple Concurrent Ejects

To minimize the library hardware time required to complete an eject, organize the cartridge tapes to be ejected by LSM. Include a CAP in that LSM with the eject command(s) to avoid pass-throughs. To minimize the operator time required to remove ejected cartridges, select a CAP closest to the operator station.

3. CSCUI CONTROL FUNCTIONS

This chapter describes the following CSCUI control functions:

- REGISTER (see page 3-6)
- DEREGISTER (see page 3-9)
- TERMINATE-REQUEST (see page 3-11)

The register informs CSCUI that a user program wants to use the user requests and notification exits. The user program should deregister from CSCUI on exit.

Each control function's discussion includes:

- a brief description of the control function
- a figure showing the packet definitions for that control function
- a table describing each of the fields in that control function's packet

In each of the figures and tables, parameter names in **bold** text indicate parameters returned; parameter names in normal text indicate parameters passed. A checkbox (☑) to the left of a field name indicates a required field.

CALLING CSCUI CONTROL FUNCTIONS

This section describes the MASM calling interface as well as the UCS C and UCOB calling interface.

MASM Calling Interface

Because CSCUI code resides in a common bank, you issue CSCUI control functions using the following code sequence:

```
LXI,U A0,packet-length
LXM,U A0,packet-address
LXI,U X11,CSCUI-BDI
LIJ   X11,01000
```

- The value of the CSCUI BDI is defined at the site when CSCUI is configured.
- Control always returns to the instruction following the LIJ instruction.
- *Packet-length* and *packet-address* define the size and location of the request packet. The packet must reside in the main D-Bank.
- The activity that calls CSCUI must use the major register set.

The following code sequence uses a proc to generate the CSCUI call:

```
CSCUI packet-length,packet-address
```

To use this proc, the program must have the following statement:

```
$INCLUDE 'SYS$LIB$*CSC-2.USERCSCUIDEF'
```

The following register changes occur when CSCUI processes control functions.

Table 3-1. Registers Used by CSCUI Control Functions

Register	Description
X11	Returns address.
A0	Contains the user request packet length and address. The length may be changed during the processing of an ACCEPT or ACCEPTW function.
A1	Returns the CSCUI-STATUS found in the user request packet.
A2, R1	Used for internal CSCUI processing.

UCS C and UCOB Calling Interface

The extended mode interface to the basic mode CSCUI bank can be used by both C and COBOL. Four C calling sequences are provided as well as two COBOL interfaces.

The C calling sequences are as follows:

1. `status = UCS$CSCUI1(pkt_sz, &pkt);`
2. `status = UCS$CSCUI2(&pkt_sz, &pkt);`
3. `status = UCS$CSCUI3(&pkt_sz, &pkt, &stat);`
4. `status = UCS$CSCUI4(&pkt_sz, &pkt, &stat, &exits);`

where:

`status` is the status returned from the CSCUI or one of the following status values if the interface detected an error before calling CSCUI:

-1 invalid number of parameters
-2 invalid parameter format or size
-3 invalid CSCUI BDI configured

`pkt_sz` is the size of the CSCUI packet in words, passed and returned.

`pkt` is the CSCUI packet.

`stat` is the UI status that is returned from the UI call.

`exits` is a bit-mapped value containing the configured exits.

The COBOL calls are:

1) `CALL 'UCS$CSCUI' USING pkt-len, pkt, stat.`
2) `CALL 'UCS$CSCUI' USING pkt-len, pkt, stat, exits.`

where:

`pkt-len` is the PIC 9(10) binary length in words.

`pkt` is the CSCUI packet definition.

`stat` is the PIC 9(10) binary CSCUI status.

`exits` is the PIC 1(36) binary bit-mapped value of the configured exits.

See Appendix B, “Sample Programs,” for a more detailed discussion of the extended mode CSCUI interface.

NOTE

The extended mode interface is not designed for simultaneous use by multiple activities. If you have a multi-activity program, only one activity can use this interface.

CSCUI CONTROL FUNCTION RESTRICTIONS

When using CSCUI control functions, you should be aware of the following restrictions.

CSCUI Services

If you have a run that receives AFTER-MOUNT and AFTER-DISMOUNT notifications, it should start *immediately* after CSC starts.

If mount activity is high, delaying the start of the user program might exhaust the CSCUI common bank queues. If this happens, some AFTER-MOUNT and AFTER-DISMOUNT would not be passed to your program. They would, however, be available in the system log.

CSCUI reports only OS 2200-initiated cartridge tape mounts. If an operator performs a MOUNT or DISMOUNT command via the server console, the activity is not reported by CSCUI notifications. This restriction also applies to requests to the NCS library server sent by *CSC CMD keyins.

VALID-FIELDS-PASSED Field

The CSCUI common bank compares the VALID-FIELDS-PASSED field to the valid fields allowed for the requested control functions, user requests, and notification exits. The only fields CSCUI validates are the FUNCTION and EXIT-ID fields.

WAIT-TIME and VALID-FIELDS-RETURNED Fields

The CSCUI common bank uses the WAIT-TIME field to time the user request. This field is not marked as VALID-FIELDS-RETURNED, and may not contain the same value when the packet returns from the common bank. *Users must refresh this field before each call.*

Reloading the Common Bank

Do not reload the common bank while CSC is up, or while user programs are using CSCUI. If you do:

- Data and requests may be lost
- Registered user programs may not function properly
- CSC may stop processing user requests or providing notification exit data

ADDRESSING CSCUI CONTROL FUNCTIONS

Keep the following addressing restrictions in mind when using CSCUI control functions:

- The CSCUI common bank occupies addresses 01000 through 077777. This is a guaranteed entry common bank, always entered at address 01000.

- The "LIJ" instruction used to enter CSCUI must base the bank on the Main I-Bank Descriptor Register (BDR). Basing the CSCUI bank on any other BDR causes a guard mode contingency.

- The CSCUI request packet must reside in a bank based on the Main D-Bank BDR, and must have a starting address of 0100000 or greater. The bank cannot be write-protected.

Failure to comply with any of these conditions produces a status of "INVALID-PACKET-ADDRESS," returned in register A1.

USING CSCUI

There are two functional groups within CSCUI:

- CSCUI User Requests
- CSCUI Notification Exits

User requests pass control requests to the server. They define a function to be performed (e.g., VOLUME-INFORMATION, SCRATCH/UNSCRATCH).

Notification exits pass information regarding a particular function back to the requester (e.g., MOUNT, DISMOUNT, ENTER, EJECT).

During a session with CSCUI, user programs use the following functions in sequence:

- REGISTER
- Any number of data transfer functions described in Chapters 3, 4, and 5
- DEREGISTER

When using user requests and notification exits, a program must first REGISTER with CSCUI. The REGISTER control function enables CSCUI to perform program-related initialization. This control function also informs CSCUI which notification exits and user requests the program will use.

After the REGISTER is done, programs can use CSCUI data transfer functions to receive data from notification exits, or to submit user requests.

To terminate properly, the user program must DEREGISTER with CSCUI at the end of processing.

CSCUI Packet Formats

Prior to CSC release 2R5, CSCUI packets had a different format. This level of CSCUI supports the pre-2R5 packet format. The older packet format is no longer documented and not recommended for new programs.

CONTROL FUNCTION DESCRIPTIONS

Each control function’s discussion includes:

- a brief description of the control function
- a figure showing the packet definitions for that control function
- a table describing each of the fields in that control function’s packet

In each of the figures and tables, parameter names in **bold** text indicate parameters returned; parameter names in normal text indicate parameters passed. A checkbox (☑) to the left of a field name indicates a required field. A complete listing of CSCUI-STATUS codes appears in Appendix A.

REGISTER Control Function

The REGISTER control function updates internal within CSCUI tables that define known user programs. A user program must register with CSCUI, using the REGISTER control function, prior to any other calls.

0	☑EXIT-ID	**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2	**UI-LEVEL-INFO**		
3	**CSC-LEVEL-INFO**		
4	RESERVED		
5	☑VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			☑OPTIONS

Figure 3-1. REGISTER Control Function Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 3-2. REGISTER Control Function Field Descriptions

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>☑ EXIT-ID</td><td colspan="2">The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.</td></tr>
<tr><td rowspan="9"><b>CSCUI-STATUS</b></td><td colspan="2">Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the user request.</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>User request completed normally.</td></tr>
<tr><td>3</td><td>EXIT-ID field specified an invalid notification exit or user request identifier.</td></tr>
<tr><td>4</td><td>User request not accepted, or not processed, because:
• invalid function
• not enough fields specified
• too many fields specified
• wrong fields specified
• no exits supplied</td></tr>
<tr><td>8</td><td>The maximum number of users are already REGISTERed for one of the indicated user request interfaces.</td></tr>
<tr><td>11</td><td>CSC is not available.</td></tr>
<tr><td>15</td><td>The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.</td></tr>
<tr><td>21</td><td>Wrong version specified with new interface format.</td></tr>
<tr><td>☑ NIF INDICATOR</td><td>99</td><td>New interface format indicator.</td></tr>
<tr><td>☑ VERSION</td><td>1</td><td>Identifies the packet format version.</td></tr>
<tr><td>☑ FUNCTION</td><td>6</td><td>REGISTER</td></tr>
<tr><td><b>UI-LEVEL-INFO</b></td><td colspan="2">CSCUI internal level in quarter word binary format:
Q1 - Major product level
Q2 - Minor product level
Q3 - Internal product level
Q4 – Unused</td></tr>
<tr><td><b>CSC-LEVEL-INFO</b></td><td colspan="2">CSC internal level in quarter word binary format:
Q1 - Major product level
Q2 - Minor product level
Q3 - Internal product level
Q4 – Unused</td></tr>
</table>

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>☑ VALID-FIELDS-PASSED</td><td colspan="2">REQUIRED: 000 000 000 400
OPTIONAL: 000 000 000 000</td></tr>
<tr><td><b>VALID-FIELDS-RETURNED</b></td><td colspan="2">REQUIRED: 020 000 000 000
OPTIONAL: 000 000 000 000</td></tr>
<tr><td><b>TIME STAMP</b></td><td colspan="2">Time, in DWTIME$ format, when CSCUI processed each user request.</td></tr>
<tr><td rowspan="5">☑ OPTIONS</td><td colspan="2">Field indicating disposition of data queued to the indicated notification exit.</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>Data queued to the indicated notification exit is not affected.</td></tr>
<tr><td>1</td><td>Instructs CSCUI to discard data already queued to the indicated notification exit interface.</td></tr>
<tr><td>2</td><td>Instructs CSCUI to discard data queued to the indicated notification exit when this user DEREGISTERs from CSCUI.</td></tr>
</table>

DEREGISTER Control Function

This control function notifies CSCUI that the user program wants to terminate its use of CSCUI requests and notifications.

The DEREGISTER control function disables notification exits registered by the requester, and reverses the REGISTER control function. In addition, the following occurs for each registered notification exit:

- CSCUI returns status 17 (“Program has DEREGISTERED”) to any outstanding ACCEPTW (ACCEPT Wait) or SENDW (SEND Wait) control functions issued by the DEREGISTERing requester. This indicates that the program issued a DEREGISTER control function.

- Notification exit queue entries destined for this run are discarded, if requested by OPTIONS.

0		**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2	n/a		
3	RESERVED		
4	RESERVED		
5	☑VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			☑OPTIONS

Figure 3-2. DEREGISTER Control Function Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 3-3. DEREGISTER Control Function Field Descriptions

Field	Description	
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the user request.	
	Value	**Description**
	0	User request completed normally.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	7	REGISTER function was not performed.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
	21	Wrong version specified with new interface format.
☑ NIF INDICATOR	99	New interface format indicator.
☑ VERSION	1	Identifies the packet format version.
☑ FUNCTION	3	DEREGISTER
☑ VALID-FIELDS-PASSED	REQUIRED: 000 000 000 400 OPTIONAL: 000 000 000 000	
VALID-FIELDS-RETURNED	n/a	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
☑ OPTIONS	Field indicating disposition of data queued to the indicated notification exit.	
	Value	**Description**
	0	Data queued to the indicated notification exit is not affected.
	1	Instructs CSCUI to discard data queued to the indicated notification exit when this user DEREGISTERs from CSCUI.

TERMINATE-REQUEST Control Function

This control function terminates a “wait for reply” (ACCEPTW or SENDW control function).

The TERMINATE-REQUEST control function returns control to an activity that is waiting with an ACCEPTW or SENDW control function. The TERMINATE-REQUEST and waiting control functions must be issued by the same run.

Terminating a SENDW control function returns control to the program that issued the SENDW function. It does not prevent the requested action from being completed. The actual user request may have already been sent to the server for processing.

0		**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2	n/a		
3	RESERVED		
4	RESERVED		
5	☑VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			
10	☑USER-SUPPLIED-REQUEST-ID		

Figure 3-3. TERMINATE-REQUEST Control Function Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 3-4. TERMINATE-REQUEST Control Function Field Descriptions

Field	Description	
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the user request.	
	Value	**Description**
	0	User request completed normally.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	5	REQUEST-ID was not supplied.
	7	REGISTER control function was not performed.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
	16	No waiting user request found with the indicated REQUEST-ID.
	21	Wrong version specified with new interface format.
☑ NIF INDICATOR	99	New interface format indicator.
☑ VERSION	1	Identifies the packet format version.
☑ FUNCTION	10	TERMINATE-REQUEST
☑ VALID-FIELDS-PASSED	REQUIRED: 000 100 000 000 OPTIONAL: 000 000 000 000	
VALID-FIELDS-RETURNED	n/a	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
☑ USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify the previous ACCEPTW or SENDW control function the user wants to terminate.	

4. CSCUI USER REQUESTS

This chapter describes the CSCUI user requests:

- DO-ENTER (see page 4-2)
- DO-VOLRPT (see page 4-6)
- INITIATE-EJECT (see page 4-17)
- MOUNT (see page 4-23)
- SCRATCH/UNSCRATCH (see page 4-30)
- VOLUME-INFORMATION (see page 4-35)

Each explanation includes:

- Brief description of the user request
- Figure showing the packet definitions for that user request
- Table describing each of the fields in that user request's packet

In each of the figures and tables, parameter names in **bold** text indicate parameters returned; parameter names in normal text indicate parameters passed. A checkbox (☑) to the left of a field name indicates a required field.

A complete listing of CSCUI-STATUS codes appears in Appendix A.

USER REQUEST PROTOCOL

CSCUI user requests allow a user program to pass processing requests to CSC and the server. The general calling protocol is:

- REGISTER
- SENDW for user request (*repeat as necessary*)
- DEREGISTER

A user program can submit one or more SENDW (SEND Wait) requests to CSCUI for processing. CSC accepts the SENDW request packet, forwards the request to the server for processing, and returns the reply in the original SENDW packet. The user program that issued the SENDW request is suspended during the processing.

DO-ENTER USER REQUEST

The DO-ENTER user request starts an operation on the server to enter cartridge tapes into the ACS. The user request is complete after the operator opens and closes the CAP, and the server enters the volumes into the ACS. If no tapes are placed in the CAP, the enter operation still completes normally.

The user can specify the LSM to use for the ENTER operation in the packet. If the user does not specify an LSM, the server will select the LSM. The user can not specify a specific CAP. The server always selects the CAP within the LSM.

CSCUI sends the DO-ENTER packet to CSC. CSC sends the ENTER request to the server for processing. The operator enters the tapes into the CAP. When the server completes the ENTER request, a response is sent to CSC. CSC generates an AFTER-ENTER notification for each cartridge tape in the response. Then CSC returns a response to the DO-ENTER request, and the process completes.

Keep in mind that while multiple user programs can register for the DO-ENTER user request, only one user program can register for the AFTER-ENTER notification exits. Therefore, only one user program can receive the notification exit data.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9				
10	USER-SUPPLIED-REQUEST-ID			
11				
12	USER-LSM			WAIT-TIME
13				
14				**REQ-COMPL-STAT**
15	LSM-ID-1			

Figure 4-1. DO-ENTER User Request Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 4-1. DO-ENTER User Request Field Descriptions

	Field	Description	
☑	EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
		Value	**Description**
		7	CSC notification exit(s) to which each DO-ENTER user request is sent.
	CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the user request.	
		Value	**Description**
		0	User request completed normally.
		1	User request rejected because the queue for the request was full.
		2	REQUEST-ID in the user request duplicates one from a currently active user request.
		4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
		7	REGISTER control function was not performed.
		10	CSC DEREGISTERed from CSCUI without returning data for the user request.
		12	TERMINATE-REQUEST control function terminated this user request.
		13	User request timed out without receiving data from CSC.
		21	Wrong version specified with new interface format.
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 003 100 000 000	
	VALID-FIELDS-RETURNED	REQUIRED: 000 000 040 000 OPTIONAL: 000 000 000 000	

Field	Description	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
USER-LSM	0 - 1	• If 0 is specified, ACSLS or NCS will select the LSM and CAP to use. • If 1 is specified, ACSLS or NCS will use the LSM specified in the LSM-ID-1 field.
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 returns. If this field is not supplied or is 0, CSC uses an internal timeout value. If this time elapses without receiving a response, a status of 16 is returned in request-completion-status.	
REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
	Value	**Description**
	0	User request completed normally.
	1	Server error occurred for which there is no existing status.
	6	ACS or LSM is invalid or does not exist in the library.
	8	Invalid USER-LSM value.
	9	No CAP with a nonzero priority is available in the requested LSM.
	16	The operation did not complete within the CSC internal time period.
	18	The LSM or CAP is OFFLINE.
LSM-ID-1	LSM to use to enter the cartridge tapes. This is a binary field with the first nine bits (quarter word) specifying the ACS (0 - 256), and the next nine bits specifying the LSM (0 - 15). ACSLS or NCS will select the CAP to use.	

DO-VOLRPT USER REQUEST

The DO-VOLRPT user request creates a file containing server information about volumes in the ACS. The data in the output file is based on parameters specified in an input file. DO-VOLRPT requests use CSCUI exit 6.

There is a working example of a program that requests a volume report in the CSCUI information file. This file is typically named SYS$LIB$*CSC-2.

The CSCUI DO-VOLRPT request packet contains the qualifier and filename of the input parameter file. Volrpt parameters and other report related information, including the output filename, are passed to CSC as data images in the parameter file.

A valid signon to the library server is needed to run the volrpt utility. The site-defined CSC configuration provides this information.

To process the DO-VOLRPT user request, CSC performs the following:

- Assigns the parameter file specified in the CSCUI packet, reads and processes the input parameters, and frees the parameter file.
- Assigns the file into which the volrpt output will be placed.
- Creates a TELNET session to the server and executes the volrpt utility.
- Writes the output from the volrpt to the output report file.
- Closes and frees the output report file.

Volume Report Parameters

The volume report parameters are text statements in the parameter file. These are created by the DO-VOLRPT requester. This parameter file contains one or more of the following statements:

Table 4-2. Volume Report Parameters

Statement Type	Required	Availabe on NCS	Description
ACS	No	Yes	Lists the ACSs to be included in the volume report. Only one of the ACS or LSM statements is allowed.
FORMAT	No	No	Defines the fields to be included in the volume report.
HEADINGS	No	Yes	Enables or disables page headers in the volume report.
LIST	No	Yes	Controls the types of images that CSC adds to the output file.
LSM	No	Yes	Lists the LSMs to be included in the volume report. Only one of the ACS or LSM statements is allowed
OUTPUT FILE	Yes	Yes	Gives assignment information for the file into which the report will be written.
REPORT TYPE	No	Yes	Specify the type of volume report to create.
SHOW	No	No	Requests that absent and ejected volumes be included in the volume report.
SORT	No	Yes	Tells the sort order for the volume report.
ZERO	No	No	Enables or disables leading zero suppression in the volume report.

Each parameter statement can be up to 132 characters long. Each parameter statement must fit on a single image.

ACS Statement

The ACS statement has the following format:

```
ACS acsnum ((acsnum) ...)
```

Where

- *acsnum* is the number of the ACS as known on the library server. The Volume Report utility on the NCS server only accepts a single ACS specification. If more than one ACS is specified, only the first one is accepted.

The default is ACS 0.

Examples

```
ACS 5
```

```
ACS 0 1 2
```

NOTES

Multiple ACS numbers are delimited by spaces.

Multiple ACSs are not supported on the NCS server.

FORMAT Statement (not available on NCS)

The FORMAT statement has the following format:

```
FORMAT field_name(,field_width(,spacing)) ...
```

Where

- `field_name` is the name of a report field as defined on the Solaris-based server.
- `field_width` is the number of character spaces allotted to field_name in the generated report. The server defined default size is used if this is omitted.
- `spacing` is the number of blank spaces between field_name and the next field in the generated report. Two spaces are used if this value is omitted.

Multiple FORMAT statements may be used. The fields on all FORMAT statements are treated as if they appeared sequentially on a single statement.

Example

```
FORMAT VOLUME_ID VOLUME_TYPE,1,1 POOL_ID
```

NOTE

Multiple field definitions are delimited by spaces.

The following table shows the fields and their default widths.

Table 4-3. FORMAT Statement Fields

Field Name	Field Width	Description
ACCESS_COUNT	5	The number of times the volume has been mounted.
ACCESS_DATE	20	The last access date for the volume.
CELL_ID	15	Home storage location for the volume. (ACS,LSM,panel,row,column)
DRIVE_ID	10	Drive location if the volume is mounted. (ACS,LSM,panel,drive)
ENTRY_DATE	20	Date when this volume was entered into the library data base.
LABEL_ATTR	5	Type of external label on the volume. (external or virtual)
LOCK_ID	5	Lock identifier if this volume is locked.
LOCK_TIME	20	Time and date when this volume was locked.
MEDIA_TYPE	7	The type of media identified by the external label: 3480 18 track or 36 standard 3490E 36 track long DD3A Helical scan 10 GB DD3B Helical scan 25 GB DD3C Helical scan 50 GB DLTIII DLT7000 10 GB DLTIIIXT DLT7000 15 GB DLTIV DLT7000 20 GB
OWNER_ID	20	Identity of the volume owner if the volume is owned.
POOL_ID	5	Server scratch pool associated with this volume.
VOLUME_ID	6	The volume id.
VOL_STATUS	17	Volume status as known to the server. (VOLUME_HOME, VOLUME_IN_DRIVE, etc)
VOLUME_TYPE	4	Status of volume as known to the server. (C=cleaning cartridge, D=data cartridge, S=scratch)

NOTE

Beginning with ACSLS or NCS 5.3, the format of the date fields is changed to include the century.

HEADINGS Statement

The HEADINGS statement has the following format:

```
HEADINGS state
```

Where

- `state` is either ON for headings in the report or OFF for no headings in the report.

The default is HEADINGS ON.

Example

```
HEADINGS OFF
```

NOTE

Headings are produced by the software on the library server. Since the DO-VOLRPT output file is not a print file, these headings are treated as data that are not guaranteed to appear at the top of each printed page.

LIST Statement

The format of the LIST statement is:

```
LIST what
```

Where

- `what` is SHORT, LONG, or NONE and tells CSC what information is included in addition to the volume report. SHORT tells CSC to list the parameter file contents in the output file. LONG produces a list of the defaulted parameters used in addition to listing the user specified parameter file contents. NONE produces no CSC-generated informational messages in the output file.

The default is LIST LONG.

Example

```
LIST NONE
```

NOTE

All CSC generated images in the output file begin with the characters ###. Errors detected by CSC during the processing of a DO-VOLRPT request are always added to the output file, even when LIST NONE is specified.

LSM Statement

The format of the LSM statement is:

```
      LSM acsnum,lsmnum  (acsnum,lsmnum ...)
```

Where

- *acsnum* is the number of the ACS as known by the Solaris-based server.
- *lsmnum* is the LSM number within the ACS as known by the Solaris-based server.

The volume report utility on the NCS server only accepts a single ACS specification. If an LSM statement includes LSMs in more that one ACS, only the LSMs in the initially specified ACS will be executed.

There is no default for this statement. The ACS statement default is used.

Example

```
      LSM 2,0
      LSM 1,0 1,1 1,3
```

NOTE

Multiple LSM specifications are delimited by spaces.

OUTPUT FILE Statement

The format of the OUTPUT FILE statement is:

```
   OUTPUT FILE ASG,options filename.
```

Where

- *options* are options that CSC will use on the internal @ASG of the file.
- *filename* is the qualifier, filename, fcycle, read key, and write key of the volume report output file. The filename component is mandatory. All other components are optional. OS 2200 default rules are used for any omitted fields.

There is no default for this statement.

Example

```
      OUTPUT FILE ASG,A MY*VOLRPT.
      OUTPUT FILE ASG,CP NEW*FILE(+1).
```

NOTE

Output files created with the C or U option have the project, account, and security attributes of the CSC run. This may affect the requester’s ability to access the file.

REPORT TYPE Statement (only available on NCS)

The REPORT TYPE statement has the following format:

```
REPORT TYPE type
```

Where

- `type` is the type of report to produce. It is either VOLUME or VIRTUAL for a report from a conventional ACS or from a VSM respectively.

The default is VOLUME.

Example

```
REPORT TYPE VIRTUAL
```

SHOW Statement (not available on NCS)

The SHOW statement has the following format:

```
SHOW ABSENT (ONLY)
```

Where

- `ONLY` requests that the volume report contain only volumes that are absent or ejected. To accomplish this, the volume report produced by SHOW ABSENT is filtered on the server using `egrep` to retain only volumes that are not resident.

There is no default. Absence of a SHOW statement produces a volume report that includes only volumes that are present in the requested ACS.

NOTE

Although absent and ejected volumes are not in any ACS, their location in the volume report is reported as in ACS 0 and in an undefined LSM. A volume report that requests a specific LSM or an ACS other than 0 will not include absent or ejected volumes.

SORT Statement

The format of the SORT statement is:

```
SORT (BY) what
```

Where

- `what` is VOL or VOLUME to sort by volume id or LOC or LOCATION to sort by location within the ACS.

The default is SORT BY VOLUME.

Example

```
SORT BY VOLUME
```

ZERO Statement (not available on NCS)

The format of the ZERO statement is either:

```
ZERO what

ZEROFILL what
```

Where

- `what` is either ON to zerofill numeric fields to the left or OFF to spacefill numeric report fields.

The default is ZEROFILL OFF.

Example

```
ZEROFILL OFF
```

Example Input Parameter File

The following is a DO-VOLRPT parameter file.

```
List none
HEADINGS OFF
Format Volume_ID POOL_ID volume_type MEDIA_type
ACS 1
Output File Asg,CP Test*Output(+1).
```

This produces a volume report in a new cycle of the file TEST*OUTPUT. Volumes in ACS 1 will be reported in alphabetical order. Images in the output file would be similar to the following:

```
AS0010  5     D    3480
AS0011  6     D    3480
AS0012  5     S    3480
AV0010  19    D    3490E
AV0010  7     S    3490E
AV0011  19    D    3490E
CLN100  0     C    3480
```

Request Packet

0	☑EXIT-ID	**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2	n/a		
3	RESERVED		
4	RESERVED		
5	VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			
10	USER-SUPPLIED-REQUEST-ID		
11			
12			WAIT-TIME
13			
14			**REQ-COMPL-STAT**
15			
16			
17			
18			
19	☑PARAMETER-FILE-QUALIFIER		
20			
21			
22	☑PARAMETER-FILENAME		
23			
24	PARAMETER-FILE-FCYCLE		

Figure 4-2. DO-VOLRPT User Request Packet

Table Legend

Bold text	returned parameter
Normal	passed parameter text
☑	required field

Table 4-4. DO-VOLRPT User Request Field Descriptions

	Field	Description	
☑	EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
		Value	**Description**
		6	CSC notification exit(s) to which each DO-VOLRPT user request is sent.
	CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
		Value	**Description**
		0	User request completed normally.
		1	User request was rejected because the queue for the request was full.
		2	REQUEST-ID in the user request duplicates one from a currently active user request.
		4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
		7	REGISTER control function was not performed.
		10	CSC DEREGISTERed from CSCUI without returning data for the user request.
		12	TERMINATE-REQUEST control function terminated this user request.
		13	User request timed out without receiving data from CSC.
		21	Wrong version specified with new interface format.
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 000 006 000 OPTIONAL: 001 100 010 000	
	VALID-FIELDS-RETURNED	BASE: 000 000 040 000 OPTIONAL: 000 000 000 000	

Field	Description	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. If this field is not supplied or is 0, CSC uses an internal timeout value. If this time elapses without receiving a response, a status of 16 is returned in request-completion-status.	
REQUEST-COMPLETION-STATUS	Status returned from CSC showing how the user request was processed.	
	Value	**Description**
	0	Request completed with no errors detectable by CSCTM.
	7	CSCTM terminated normally before completing the request.
	15	The connection to the server used for DO-VOLRPT output was closed by a communication error condition.
	16	During exchanges for creating DO-VOLRPT output, an expected server response was not received in the allotted time.
	17	An error occurred in creating or writing to the DO-VOLRPT output file.
	19	The CSC configuration is not properly set up to DO-VOLRPT requests. If you specified “LIST LONG” in your parameter file, then an explanation of the failure will be included in your output file. Contact your site administrator to correct the problem.
	21	An error occurred in assigning or reading the parameter file for DO-VOLRPT. No output file will be produced.
	22	An error was detected in the parameter file contents. If the OUTPUT FILE statement was valid, then the output file contains the parameters and associated error messages.
☑ PARAMETER-FILE-QUALIFIER	Qualifier of the input parameter file.	
☑ PARAMETER FILENAME	Name of the input parameter file.	
PARAMETER-FCYCLE	F-cycle of the input parameter file.	

INITIATE-EJECT USER REQUEST

The INITIATE-EJECT user request ejects cartridge tapes from an ACS. You can eject up to 42 cartridge tapes in a single packet via the INITIATE-EJECT user request.

When using the INITIATE-EJECT user request, keep in mind the following:

- Each INITIATE-EJECT request uses, at most, one CAP per LSM. To take advantage of LSMs with multiple CAPs, issue multiple INITIATE-EJECT requests.
- While multiple user programs can register for the INITIATE-EJECT user request, only one user program can register for the AFTER-EJECT notification exit and EJECT-COMPLETION notification exit. Therefore, only one user program can receive the notification exit data.
- If a CAP was not specified on the eject command, then the server software will select the highest non-zero priority CAP and use it for the duration of that eject.
- Site specific settings determine what action is taken. If the specified CAP is not available or if the number of concurrent ejects is greater than the number of available CAPs, the eject request is either queued in CSC until the CAP is available or the request terminates with a request completion status of 9.

Using INITIATE-EJECT User Requests

The INITIATE-EJECT user request ejects cartridge tapes from the ACS in the following sequence:

1. The SENDW control function issues an INITIATE-EJECT user request. The user can then specify an LSM and CAP from which to eject the cartridge tapes. If the user doesn’t specify an LSM and CAP, the library server software will select the LSM and CAP(s). If the user specifies an LSM but not a CAP, the server software will select the CAP(s).
2. CSC receives the INITIATE-EJECT user request. CSC then validates that the USER-LSM, VOLUME-COUNT, LSM-ID, and CAP are within range, and that the volsers contain valid characters.
   - If there are any errors, CSC returns an error response to the INITIATE-EJECT user request, and the process completes.
   - If there are no errors, CSC sends the INITIATE-EJECT user request to the server for processing.
3. CSC returns a response to the INITIATE-EJECT user request under the following conditions:

- The request was accepted by the library server and the eject request did not specify a timeout value and CSC was configured by the site to not queue any eject responses. CSC provides a unique identifier for the user request in the RETURNED-REQUEST-ID field. The user program can then associate subsequent AFTER-EJECT and EJECT completion notifications with the original INITIATE-EJECT user request.

- The request was rejected by the library server because the required CAP was not available and CSC was not configured by the site to not queue any eject responses.

4. When the server completes the tape movement and the operator has removed all of the ejected cartridge tapes from the CAP, the server sends a response to CSC.

5. If the INITIATE-EJECT user request did not specify a timeout value, and the server response indicates a retryable condition, then the INITIATE-EJECT request is queued in CSC and retried later.

6. CSC generates the following:

   - A response to the INITIATE-EJECT user request is returned if one was not already returned in Step 3.

   - An EJECT-COMPLETION notification that contains the RETURNED-REQUEST-ID is sent.

   - An AFTER-EJECT notification is sent for each cartridge tape in the response. Each notification exit contains the RETURNED-REQUEST-ID.

NOTE

Remember that operator intervention is required to unload cartridge tapes from the CAP before the eject is considered to be completed.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9	☑VOLUME-COUNT			RETURND-EXIT-ID
10	☑USER-SUPPLIED-REQUEST-ID			
11	**RETURNED-REQUEST-ID**			
12	USER-LSM			WAIT-TIME
13				
14				**REQ-COMPL-STAT**
15	LSM-ID-1			
16				
17			CAP-NUMBER	
18	☑VOLUME-ID-1			
19				
20	VOLUME-ID-2			
21	VOLUME-ID-3			
22				
23	VOLUME-ID-4			
	. . .			
	VOLUME-ID-41			
	VOLUME-ID-42			

Figure 4-3. INITIATE-EJECT User Request Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 4-5. INITIATE-EJECT User Request Field Descriptions

	Field	Description	
☑	EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
		Value	**Description**
		8	CSC notification exit(s) to which each INITIATE-EJECT user request is sent.
	CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the user request.	
		Value	**Description**
		0	User request completed normally.
		1	User request rejected because the queue for the request was full.
		2	REQUEST-ID in the user request duplicates one from a currently active user request.
		4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
		7	REGISTER control function was not performed.
		9	Control function is illegal for the requested notification exit(s).
		10	CSC DEREGISTERed from CSCUI without returning data for the user request.
		12	TERMINATE-REQUEST control function terminated this user request.
		13	User request timed out without receiving data from CSC.
		21	Wrong version specified with new interface format.
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 700 000 000 OPTIONAL: 043 000 000 004	

	Field	Description	
	VALID-FIELDS-RETURNED	REQUIRED: 000 000 140 000 OPTIONAL: 000 100 200 000	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
☑	VOLUME-COUNT	1 - 42	Number of cartridge tapes contained in this packet.
	RETURNED-EXIT-ID	The acknowledge EXIT-ID returned by CSCUI. Should be identical to the EXIT-ID specified in word 0, H1.	
☑	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
	RETURNED-REQUEST-ID	Identifier, assigned by CSC, which the server uses to refer to a specific INITIATE-EJECT user request. The value of this field matches the RETURNED-REQUEST-ID of the original INITIATE-EJECT user request. This field is only present for eject notification exits generated by CSC at the completion of an INITIATE-EJECT user request.	
	USER-LSM	0 - 1	• If 0 is specified, ACSLS or NCS will select the LSM and CAP to use. • If 1 is specified, ACSLS or NCS will use the LSM specified in the LSM-ID-1 field.
	WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving an eject completion response, a timeout CSCUI-STATUS 13 returns. If this field is not supplied or is 0, CSC sends the eject request to the server and returns control to the INITIATE-EJECT requester. The request continues asynchronously without further timing.	
	REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
		Value	**Description**
		0	User request completed normally.
		1	Server error occurred for which there is no existing status.
		6	Specified LSM, ACS, or CAP is not in library.
	REQUEST-COMPLETION-STATUS *(cont'd)*	7	User request cannot be completed because: • the user request specified a non-zero wait time, and CSC did not receive the response to the user request within the specified time, or • CSC did not receive the response to the user request within a reasonable time.

Field	Description	
	8	Indicates a parameter error for one of the following conditions: • invalid CAP-COUNT • invalid VOLUME-COUNT • invalid characters in VOLUME-ID • invalid ACS or LSM in CAP list
	9	CAP is busy or in use**.** This status may indicate one of the following conditions: • the ejected cartridge tapes have not been removed from the CAP • no non-zero priority CAP is available
	20	EJECT request is a duplicate request.
	23	Server reported a retryable condition, but CSC did not have the resources to queue the request.
LSM-ID-1	LSM to use to eject the cartridge tapes. This is a binary field with the first nine bits (quarter word) specifying the ACS (0 - 256), and the next nine bits specifying the LSM (0 - 15). ACSLS or NCS will select the CAP to use, if CAP-NUMBER is not specified.	
CAP-NUMBER	CAP within the specified LSM-ID-1 to use to eject the cartridge tapes.	
☑ VOLUME-ID-*n*	• Cartridge tape to be ejected. This field is repeated VOLUME-COUNT times.	

MOUNT USER REQUEST

The MOUNT user request sends a MOUNT cartridge tape request to the server.

For each MOUNT request, the processing code performs the following operations:

- CSC ensures that the request is for a known transport. If this verification fails then a parameter error status is returned in the request-completion status.

- If needed, CSC translates the CTL-pool name and label type into a server scratch pool identifier. A ‘CTL-POOLNAME does not exist’ status is returned in the request-completion status if this translation cannot be done, and the UNDEFINED_POOL MOUNT ACTION parameter in the CSC configuration is set to REJECT.

- CSC then creates an ACSLS or NCS mount request and submits it to the library server for processing.

- If the library server returns a failure status for which CSC has a recovery process, then CSC does the recovery actions.

- The request and recovery of the previous two steps repeat until one of the following events occurs. Then CSC reports the indicated server status in the returned-server-status field.

  - The mount completes successfully. CSC reports a successful mount.

  - The time allowed for the mount expires and the request times out. CSC reports a failure status in the request-completion status and an NI_TIMED_OUT server status.

  - The server returns a status for which CSC has no recovery process. CSC reports a failure status in the request-completion status and includes the unrecoverable server status.

- If the mount completes successfully, CSC retains a copy of any identifying user information in the mount request. This information will be included in CSCUI AFTER-MOUNT and AFTER-DISMOUNT notifications.

A failure status is also returned if the CSCUI MOUNT request specified a non-zero wait time and the request cannot be processed immediately because the transport is busy doing another ACS operation. In this case the returned-server-status will contain a DRIVE-IN-USE status. This is done because request timing is not possible in CSC while a request is on the pending queue waiting for a transport to complete an operation. The CSCUI timing could therefore not be honored.

The fields in the MOUNT request packet permit several types of mounts. A scratch mount is indicated when the volume-id field contains 0, or one of the following ASCII values: “   ” (6 blank spaces), “@@@@@@” (6 @’s), “______” (6 underscores), and “BLANK ” (the word “BLANK” followed by 1 space). The label type is indicated in the tape options field for scratch mounts, regardless of the value in volume-id. The following table shows the types of mounts along with the corresponding MOUNT request packet field requirements.

Table 4-6. Mount Types and Request Packet Field Requirements

Type of Mount	Packet Field	Description
Specific mount	VOLUME-ID	Presence of the VOLUME-ID field indicates a specific mount request.
	TAPE-OPTIONS	Bit 0 (low bit) -- not used Bit 1 -- set to mount tape in read only mode Bit 2 -- not used
	CTL-POOL EXT-SCRATCH-POOL	Not used
Scratch mount (default pool)	VOLUME-ID	Absence of this field indicates a scratch mount request.
	TAPE-OPTIONS	Bit 0 (low bit) -- set to request unlabeled tape Bit 1 -- should be 0 Bit 2 -- must be 0 to indicate no CTL pool
	CTL-POOL	Not allowed
	EXT-SCRATCH-POOL	Absence of this field indicates that one of the default scratch pools should be used.
Scratch mount (specific ACS pool)	VOLUME-ID	Absence of this field indicates a scratch mount request.
	TAPE-OPTIONS	Bit 0 (low bit) -- not used Bit 1 -- should be 0 Bit 2 -- must be 0 to indicate no CTL pool
	CTL-POOL	Not allowed
	EXT-SCRATCH-POOL	The ACS pool from which to mount the volume.
Scratch mount (specific CTL pool)	VOLUME-ID	Absence of this field indicates a scratch mount request.
	TAPE-OPTIONS	Bit 0 (low bit) -- set to request unlabeled tape Bit 1 -- should be 0 Bit 2 -- must be 1 to indicate CTL pool option
	CTL-POOL	CTL pool name from which to select the volume.
	EXT-SCRATCH-POOL	Not allowed

0	☑EXIT-ID	**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2			
3	CTL-POOL		
4			
5	☑VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			
10	USER-SUPPLIED-REQUEST-ID		
11	**RETURNED-SERVER-STATUS**		
12	EXTENDED-SCRATCH-POOL		WAIT-TIME
13	VOLUME-ID		
14		☑TAPE-OPTIONS	REQ-COMPL-STAT
15	☑UNIT-NAME		
16		EXPIRATION-PERIOD	
17	REL-REEL		
18			
19	QUALIFIER		
20			
21			
22	FILE-NAME		
23			
24	FCYCLE		
25	ORIGINAL-RUN-ID		
26		GENERATED-RUN-ID	
27			
28			
29	ACCOUNT-NO		
30			
31			
32	USER-ID		
33			

Figure 4-4. MOUNT User Request Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 4-7. MOUNT User Request Field Descriptions

Field	Description	
☑ EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	1	CSC notification exit(s) to which each MOUNT user request is sent.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	1	User request rejected because the queue for the request was full.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated this function.
	13	User request timed out without receiving data from CSC.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.

	Field	Description	
	CSCUI-STATUS *(cont'd)*	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
		17	The requesting program DEREGISTERed while the user request was waiting.
		21	Wrong version specified with new interface format.
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
	CTL-POOLNAME	CTL-POOLNAME to use for the mount request.	
☑	VALID-FIELDS-PASSED	REQUIRED: 000 024 000 000 OPTIONAL: 015 540 037 062	
	VALID-FIELDS-RETURNED	BASE: 000 000 040 000 OPTIONAL: 000 000 200 000	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
	RETURNED-SERVER-STATUS	The server status from the ACSLS or NCS mount request.	
		Value	**Description**
		8	An AUDIT process on the server prevents this mount at this time.
		23, 56,74	A server failure was detected by the server software.
		29	The requested drive already contains a loaded tape cartridge.
		31, 62	The requested drive or the LSM containing that drive is offline to the server.
		53	The volume is syntactically incorrect.
		55, 57	The ACS or the server software is temporarily unavailable.
		65	The external label on the tape cartridge found in the location indicated in the server database does not match the label in the server database.
		69	The tape cartridge and the tape drive in the request are in different ACSs.

Field	Description		
RETURNED-SERVER-STATUS *(cont'd)*	91	The requested tape cartridge is already mounted on another ACS drive.	
	94	The requested volume is not found in the server database of all tapes in the ACS.	
	95	The cartridge found in the location indicated in the server database has an external label that cannot be read by the ACS hardware and no virtual label has been assigned.	
	99	The requested tape cartridge is reserved for another server operation.	
	118	The pool requested for a scratch mount does not exist on the server.	
	135	No volumes were available in the requested ACS to satisfy a scratch mount request.	
	157	The requested drive is down.	
	*	Other statuses report software failures or configuration errors that should be reported to the system administrator.	
EXTENDED-SCRATCH-POOL	Scratch pool to use for the mount request.		
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. If this field is not supplied or is at 0, CSC uses an internal timeout value. If this time elapses without receiving a response, a status is returned in request-completion-status.		
VOLUME-ID	Cartridge tape to be mounted, or binary 0 if scratch mount.		
☑ TAPE-OPTIONS	A bitmapped field containing information about the mount request.		
	Bitmap	**Value**	**Description**
	000	0	Mount a labeled scratch cartridge tape.
	001	1	Mount an unlabeled scratch cartridge tape.
	010	2	Mount a tape -- READONLY
	100	4	Mount a labeled scratch tape from the pool specified by CTL-POOLNAME.
	101	5	Mount an unlabeled scratch tape from the pool specified by CTL-POOLNAME.

Field	Description	
REQUEST-COMPLETION-STATUS	Status returned from CSC showing how the user request was processed.	
	Value	**Description**
	0	User request completed normally.
	1	Server error. Additional status in returned-server-status.
	8	Parameter error.
	13	The CTL-POOLNAME does not exist in the CSC configuration.
	20	The request is a duplicate of a request already in progress or queued.
☑ UNIT-NAME	OS 2200 name of the unit on which to mount the tape.	
EXPIRATION-PERIOD	Retention time, in days, specified on the cartridge tape assignment. If no retention time was specified, defaults to the OS 2200 configured default.	
REL-REEL	Relative reel number for this cartridge tape file. This is relative to the OS 2200 file.	
QUALIFIER	Qualifier for this cartridge tape file.	
FILE-NAME	File name for this cartridge tape file.	
FCYCLE	Absolute file cycle for this cartridge tape file. This is a character field.	
ORIGINAL-RUN-ID	Run ID specified on the @RUN statement for the run requesting this cartridge tape mount.	
GENERATED-RUN-ID	Run ID used by OS 2200 to uniquely identify the run that requested this cartridge tape.	
ACCOUNT-NO	OS 2200 account number of the run that requested this cartridge tape.	
USER-ID	OS 2200 user ID of the run that requested this cartridge tape.	

SCRATCH/UNSCRATCH USER REQUEST

The SCRATCH/UNSCRATCH user request controls the scratch status of volumes known to the server. This request can either scratch or unscratch a single volume, or multiple volumes, by setting the optional VOLUME-COUNT field, setting the VOLUME-COUNT bit in the VALID-FIELDS-PASSED field, and setting the additional volumes in the VOLUME-ID-n fields. Up to 42 volumes can be scratched or unscratched in a single request.

With the ACSLS library control software, this request can also set the scratch pool ID for the indicated volume(s) by using the scratch-pool-indicator field. If the scratch-pool-indicator field is not present, only the scratch status is changed based on the scratch-indicator field. If the scratch-pool-indicator field is present, its value determines which default pool or field of the CSCUI scratch request will contain the scratch pool ID. The extended-scratch-pool-number and CTL-POOLNAME fields of the scratch packet are valid only if the scratch-pool-indicator field is present. For multiple volume requests, these fields apply to all volumes to be scratched or unscratched by this request. The following table shows the scratch-pool-indicator values corresponding to where the scratch pool ID is obtained from:

Table 4-8. Scratch Pool Indicator Value/ID Reference (ACSLS only)

Value	Scratch pool ID determined from:
0	the pool from the DEFAULT_LABELED_POOL configuration statement
1	the pool from the DEFAULT_UNLABELED_POOL configuration statement
2	the pool which maps to “poolname,LABELED” in the TRANSLATE_POOL configuration statement, where poolname is passed in the CTL-POOLNAME field
3	the pool which maps to “poolname,UNLABELED” in the TRANSLATE_POOL configuration statement, where poolname is passed in the CTL-POOLNAME field
4	the pool passed in the extended-scratch-pool-number field

NOTE

In the absence of the scratch-pool-indicator field, the scratch pool is not changed. Only the scratch status is changed.

If the scratch-indicator specifies an unscratch request and the scratch-pool-indicator field is present, the scratch pool ID will be set according to the value in the scratch-pool-indicator and the scratch designation will be cleared. Upon the next request to set the scratch designation, if the scratch-pool-indicator field is not present, the scratch pool ID from the unscratch request will be used. In this way, an unscratch request can be used to change the scratch pool ID of the indicated volume(s), without scratching the volume(s).

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	CTL-POOLNAME			
3				
4				
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9	VOLUME-COUNT			**RETURND-EXIT-ID**
10	☑USER-SUPPLIED-REQUEST-ID			
11				
12	EXTENDED-ACS-POOL-NBR		SCR-POOL-INDCTR	WAIT-TIME
13	☑VOLUME-ID			
14			☑SCRCH-INDCTR	**REQ-COMPL-STAT**
15	VOLUME-ID-2			
16				REQ-COMPL-STAT-2
17	VOLUME-ID-3			
18				REQ-COMPL-STAT-3
19	VOLUME-ID-4			
20				REQ-COMPL-STAT-4
	⋮			
	VOLUME-ID-42			
				REQ-COMPL-STAT-42

Figure 4-5. SCRATCH/UNSCRATCH User Request Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

NOTE

For multiple volume requests, the scratch-indicator, scratch-pool-indicator, extended-ACS-pool-nbr, and CTL-poolname fields apply to all volumes specified in the request.

Table 4-9. SCRATCH/UNSCRATCH User Request Field Descriptions

	Field	Description	
☑	EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
		Value	**Description**
		9	CSC notification exit(s) to which each SCRATCH/UNSCRATCH user request is sent.
	CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
		Value	**Description**
		0	User request completed normally.
		1	User request rejected because the queue for the request is full.
		2	REQUEST-ID in the user request duplicates one from a currently active user request.
		4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
		7	REGISTER control function was not performed.
		9	Control function is illegal for the requested notification exit(s).
		10	CSCAM DEREGISTERed from CSCUI without returning data for the user request.
		12	TERMINATE-REQUEST control function terminated this user request.
		13	User request timed out without receiving data from CSC.
		15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
		17	The requesting program DEREGISTERed while the user request was waiting.
		21	Wrong version specified with new interface format.
☑	NIF INDICATOR	99	New interface format indicator.

	Field	Description	
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
	CTL-POOLNAME	Scratch tape(s) to ACS pool corresponding to this CTL-pool and SCR-POOL-INDICATOR.	
☑	VALID-FIELDS-PASSED	REQUIRED: 000 500 400 000 OPTIONAL: 015 201 000 000	
	VALID-FIELDS-RETURNED	BASE: 000 000 040 000 OPTIONAL: 000 100 000 000	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
	VOLUME-COUNT	1 - 42	Number of cartridge tapes contained in this packet.
	RETURNED-EXIT-ID	9	SCRATCH/UNSCRATCH notification exit
☑	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
	EXTENDED-ACS-POOL-NBR	*(ACSLS only)* Scratch tape(s) to ACS-POOL.	
	SCR-POOL-INDICATOR	*(ACSLS only)* Directs pool for scratch request: 0 = default labeled 1 = default unlabeled 2 = labeled CTL-POOLNAME 3 = unlabeled CTL-POOLNAME 4 = ACS pool in EXTENDED-ACS-POOL-NBR	
	WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. If this field is not supplied or is 0, CSC uses an internal timeout value. If this time elapses without receiving a response, a status is returned in request-completion-status.	
☑	VOLUME-ID	Volume to be updated. For multiple volume requests, this field contains the 1st volume to be updated.	
☑	SCRATCH-INDICATOR	Indicates one of the following requested changes:	
		Value	**Description**
		1	Change volume status to SCRATCH.
		2	Change volume status to UNSCRATCH.

Field	Description	
REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed. For multiple volume requests, this field contains the status for the 1st volume.	
	Value	**Description**
	0	User request completed normally.
	1	Server error occurred of which there is no existing status.
	2	Volume specified in the user request does not reside in the indicated ACS.
	3	Specified volume was selected at the time of the user request. This status is returned only if the requested change is to scratch the volume.
	7	User request cannot be completed because: • the user request specified a non-zero wait time, and CSC did not receive the response to the user request within the specified time, or • CSC did not receive the response to the user request within a reasonable time.
	8	Invalid SCRATCH-INDICATOR or SCR-POOL-INDICATOR parameters, or fields not supplied corresponding to SCR-POOL-INDICATOR value.
	12	ACS-POOL does not exist in ACS.
	13	CTL-POOLNAME/label type does not translate to an ACS pool.
	14	Tape was scratched to same pool because supplied CTL-POOLNAME/label type did not translate to an ACS pool.
VOLUME-ID-n	For multiple volumes requests, this field contains the additional volumes to be updated, repeated VOLUME-COUNT – 1 times.	
REQUEST-COMPLETION-STATUS-n	For multiple volume requests, this field contains the status returned from CSC or the server for the nth volume. This field is repeated VOLUME-COUNT – 1 times. The values are the same as listed in the REQUEST-COMPLETION-STATUS field above.	

VOLUME-INFORMATION USER REQUEST

The VOLUME-INFORMATION user request retrieves server-related information about a cartridge tape, such as the ACS ID.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9	MEDIA-TYPE	**ACS-ID**	**LSM-ID**	**RETURND-EXIT-ID**
10	☑USER-SUPPLIED-REQUEST-ID			
11				
12				WAIT-TIME
13	☑VOLUME-ID			
14				**REQ-COMPL-STAT**
15	**UNIT-NAME**			
16				
17				
18				
19				

Figure 4-6. VOLUME-INFORMATION User Request Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 4-10. VOLUME-INFORMATION User Request Field Descriptions

Field	Description	
☑ EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	10	CSC notification exit(s) to which each VOLUME-INFORMATION user request is sent.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	1	User request was rejected because the queue for the request was full.
	2	REQUEST-ID in the user request duplicates one from a currently active user request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	7	REGISTER control function was not performed.
	9	Control function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST control function terminated this user request.
	13	User request timed out without receiving data from CSC.
	15	The request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
	17	The requesting program DEREGISTERed while the user request was waiting.
	21	Wrong version specified with new interface format.
☑ NIF INDICATOR	99	New interface format indicator.

	Field	Description	
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	9	SENDW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 500 000 000 OPTIONAL: 001 000 000 000	
	VALID-FIELDS-RETURNED	BASE: 000 000 140 000 OPTIONAL: 000 120 000 204	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
	MEDIA-TYPE	Numeric value used by the server for the media type of this volume.	
	ACS-ID	ACS in which the cartridge tape resides (if applicable).	
	LSM-ID	LSM in which the cartridge tape resides (if applicable).	
	RETURNED-EXIT-ID	10	VOLUME-INFORMATION notification exit
☑	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
	WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. If this field is not supplied or is 0, CSC uses an internal timeout value**.** If this time elapses without receiving a response, a status is returned in request-completion-status.	
☑	VOLUME-ID	Cartridge tape for which information is requested. Enter the volume serial number *exactly* as it appears on the OCR label on the cartridge tape, using the same justification and case.	
	REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
		Value	**Description**
		0	User request completed normally.
		1	Server error occurred for which there is no existing status.
		2	Cartridge tape specified in the user request does not reside in the indicated ACS.
		3	Specified cartridge tape was selected at the time of the user request.
	REQUEST-COMPLETION-STATUS *(cont'd)*	4	Specified cartridge tape is incorrect.

Field	Description	
	7	User request cannot be completed because: • the user request specified a non-zero wait time, and CSC did not receive the response to the user request within the specified time, or • CSC did not receive the response to the user request within a reasonable time.
☑ **UNIT-NAME**	OS 2200 unit name, if the cartridge tape is mounted at the time of the user request.	

5. CSCUI NOTIFICATION EXITS

This chapter describes the CSCUI notification exits:

- AFTER-DISMOUNT
- AFTER-EJECT
- AFTER-ENTER
- AFTER-MOUNT
- BEFORE-MOUNT
- EJECT-COMPLETION

Each explanation includes:

- a brief description of the notification exit
- a figure showing the packet definitions for that notification exit
- a table describing each of the fields in that notification exit's packet

In each of the figures and tables, parameter names in **bold** text indicate parameters returned; parameter names in normal text indicate parameters passed. A checkbox (☑) to the left of a field name indicates a required field.

A complete listing of CSCUI-STATUS codes appears in Appendix A.

NOTIFICATION EXIT PROTOCOL

Notification exits pass information about events in the ACS from CSC to user programs. The user programs utilize an ACCEPT or ACCEPTW (ACCEPT Wait) control function to receive this notification exit data. The notification exit calling protocol is:

- REGISTER
- ACCEPTW or ACCEPT (repeat where necessary)
- DEREGISTER

AFTER-DISMOUNT NOTIFICATION EXIT

The AFTER-DISMOUNT notification exit passes data to a user program upon completion of a CSC-initiated cartridge tape dismount. The notification exit data originates from OS 2200 at mount request time, and from the server at dismount completion time. A user program can register for the AFTER-DISMOUNT notification exit, then receive notifications whenever the library server completes a dismount request. When using this notification exit, the following considerations are important:

- The AFTER-DISMOUNT notification exit data does not specify whether the cartridge tape has been written or read or the success of any I/O operation to the tape. It tells only that a cartridge tape has been dismounted.

- Like all notification exits in CSCUI, data loss will result if you reload the CSCUI common banks.

- If you use the AFTER-DISMOUNT notification as an indicator to eject a cartridge tape, you must consider when to eject the cartridge tape, and who is attending to the CAP (since the CAP must be cleared after each eject). You must also consider the possibility that notification exits will be lost if the user program accumulates AFTER-DISMOUNT notifications before sending the eject request.

CSC converts OS 2200 UNLOAD requests to server dismount requests. Each dismount request sent to the server receives a response. Information from the OS 2200 LOAD request and the server dismount response are converted to AFTER-DISMOUNT format, then passed to CSCUI. The user program can receive AFTER-DISMOUNT notifications by using either the ACCEPT or ACCEPTW control functions. These AFTER-DISMOUNT notifications are passed through CSCUI exit number 5.

The AFTER-DISMOUNT notification exit reports only successful dismounts initiated from the client. It does not report dismounts initiated from the server console or from another client system. If CSC terminates after issuing a dismount request, but before receiving a dismount response, the dismount is not reported.

If a user job terminates in error after an UNLOAD request has been converted to an AFTER-DISMOUNT notification exit, the cartridge tape will actually dismount, but the AFTER-DISMOUNT notification exit will not detect the error termination. It is the user’s responsibility to determine the disposition of the job. In order to capture all of the relevant information for AFTER-DISMOUNT, the data must be captured at the time OS 2200 presents the UNLOAD request.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9		**ACS-ID**		RETURND-EXIT-ID
10	USER-SUPPLIED-REQUEST-ID			
11				
12				WAIT-TIME
13	**VOLUME-ID**			
14				REQ-COMPL-STAT
15	**UNIT-NAME**			
16			**EXPIRATION-PERIOD**	
17	**REL-REEL**			
18	**QUALIFIER**			
19				
20				
21	**FILE-NAME**			
22				
23				
24	**FCYCLE**			
25	**ORIGINAL-RUN-ID**			
26			**GENERATED-RUN-ID**	
27				
28	**ACCOUNT-NO**			
29				
30				
31	**USER-ID**			
32				
33				

Figure 5-1. AFTER-DISMOUNT Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-1. AFTER-DISMOUNT Notification Exit Field Descriptions

☑

Field	Description	
EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single or selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	5	CSC notification exit(s) from which each AFTER-DISMOUNT data is accepted.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated this function.
	13	User request timed out without receiving data from CSC.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
	17	The requesting program DEREGISTERed while the user request was waiting.
CSCUI-STATUS *(cont'd)*	21	Wrong version specified with new interface format.

	Field	Description	
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000	
	VALID-FIELDS-RETURNED	BASE: 000 464 177 066 OPTIONAL: 000 001 000 000	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.	
	ACS-ID	ACS in which the cartridge tape was dismounted.	
	RETURNED-EXIT-ID	5	AFTER-DISMOUNT notification exit
	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
	WAIT-TIME	Number of seconds each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the request waits indefinitely for a reply.	
	VOLUME-ID	Cartridge tape that was dismounted.	
	REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
		Value	**Description**
		0	User request completed normally.
	UNIT-NAME	OS 2200 name of the unit on which the cartridge tape was dismounted.	
	EXPIRATION-PERIOD	Retention time, in days, specified on the cartridge tape assignment. If no retention time was specified, defaults to the OS 2200 configured default.	
	REL-REEL	Relative reel number dismounted. This is relative to the OS 2200 file.	
	QUALIFIER	Qualifier for this cartridge tape file.	
	FILE-NAME	File name specified on the OS 2200 @ASG statement for this cartridge tape file.	
	FCYCLE	Absolute file cycle for this cartridge tape file. This is a character field.	
	ORIGINAL-RUN-ID	Run ID specified on the @RUN statement for the run requesting this cartridge tape dismount.	

Field	Description
GENERATED-RUN-ID	Run ID used by OS 2200 to identify the run that requested this cartridge tape.
ACCOUNT-NO	OS 2200 account number of the run that requested this cartridge tape.
USER-ID	OS 2200 user ID of the run that requested this cartridge tape.

AFTER-EJECT NOTIFICATION EXIT

The AFTER-EJECT notification exit passes information about ejected cartridge tapes to a user program. CSC generates an AFTER-EJECT notification each time the server ejects a cartridge tape in response to a CSC-initiated EJECT. A single EJECT command can produce several AFTER-EJECT notification exits.

When an operator issues an eject command from a console, or upon completion of an INITIATE-EJECT user request from a user program, CSC sends AFTER-EJECT notifications, one per cartridge tape ejected, to CSCUI. The user program receives the AFTER-EJECT information by using the ACCEPT or ACCEPTW control functions. These notification exits are passed through CSCUI exit number 2.

NOTE

There are no AFTER-EJECT notifications for cartridge tapes ejected via the library server console or from another client.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9		**ACS-ID**	**LSM-ID**	**RETURND-EXIT-ID**
10	USER-SUPPLIED-REQUEST-ID			
11	**RETURNED-REQUEST-ID**			
12				WAIT-TIME
13	**VOLUME-ID**			
14				**REQ-COMPL-STAT**

Figure 5-2. AFTER-EJECT Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-2. AFTER-EJECT Notification Exit Field Descriptions

☑

Field	Description	
EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected notification exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	2	CSC notification exit from which AFTER-EJECT data is accepted.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated the ACCEPTW control function.
	13	ACCEPTW control function timed out without receiving data from CSC.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
CSCUI-STATUS *(cont'd)*	17	The requesting program DEREGISTERed while the user request was waiting.

Field	Description	
	21	Wrong version specified with new interface format.
☑ NIF INDICATOR	99	New interface format indicator.
☑ VERSION	1	Identifies the packet format version.
☑ FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW
☑ VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000	
VALID-FIELDS-RETURNED	BASE: 000 400 140 000 OPTIONAL: 000 000 200 204	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed this notification exit.	
ACS-ID	ACS from client request (if specified).	
LSM-ID	ACS-relative LSM number from client request (if specified).	
RETURNED-EXIT-ID	2	AFTER-EJECT
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each request at a later time.	
RETURNED-REQUEST-ID	Identifier, assigned by CSC, which the server uses to refer to a specific INITIATE-EJECT user request. The value of this field matches the RETURNED-REQUEST-ID of the original INITIATE-EJECT user request. This field is only present for eject notifications generated by CSC at the completion of an INITIATE-EJECT user request.	
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the user request waits indefinitely for a reply.	
VOLUME-ID	Cartridge tape that was ejected.	
REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
	Value	**Description**
	0	User request completed normally.
	1	Server error occurred for which there is no existing status.
REQUEST-COMPLETION-STATUS *(cont'd)*	2	Cartridge tape specified in the user request does not reside in the library.

Field	Description	
	3	Specified cartridge tape was selected at the time of the user request.
	4	Specified cartridge tape is incorrect.
	6	Specified LSM, ACS, or CAP is not in the library.
	24	Specified cartridge tape does not reside in the indicated ACS.

AFTER-ENTER NOTIFICATION EXIT

The AFTER-ENTER notification exit passes information to a user program about cartridge tapes entered into the ACS. CSC generates an AFTER-ENTER notification for each cartridge tape entered via a CSC-initiated request. A single enter command can generate multiple AFTER-ENTER notifications.

When an operator issues an ENTER command from a console, or upon completion of a DO-ENTER user request from a user program, CSC sends AFTER-ENTER notifications to CSCUI, one per cartridge tape entered. The user program receives AFTER-ENTER information using either the ACCEPT or ACCEPTW control functions. These notifications are passed through CSCUI exit number 3.

NOTE

There are no AFTER-ENTER notifications for cartridges entered via commands from the server console or from another client system.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	n/a			
3	RESERVED			
4	RESERVED			
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9		**ACS-ID**	**LSM-ID**	**RETURND-EXIT-ID**
10	USER-SUPPLIED-REQUEST-ID			
11				
12				WAIT-TIME
13	**VOLUME-ID**			
14				**REQ-COMPL-STAT**

Figure 5-3. AFTER-ENTER Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-3. AFTER-ENTER Notification Exit Field Descriptions

☑

Field	Description	
EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	3	CSC notification exit(s) from which AFTER-ENTER data is accepted.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated the ACCEPTW control function.
	13	ACCEPTW control function timed out without receiving data from CSC.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
CSCUI-STATUS *(cont'd)*	17	The requesting program DEREGISTERed while the user request was waiting.
	21	Wrong version specified with new interface format.

Field	Description	
☑ NIF INDICATOR	99	New interface format indicator.
☑ VERSION	1	Identifies the packet format version.
☑ FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW
☑ VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000	
VALID-FIELDS-RETURNED	BASE: 000 400 140 000 OPTIONAL: 000 000 000 204	
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed this notification exit.	
ACS-ID	ACS from client request (if specified).	
LSM-ID	ACS-relative LSM number from client request (if specified).	
RETURNED-EXIT-ID	3	AFTER-ENTER
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.	
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the user request waits indefinitely for a reply.	
VOLUME-ID	Cartridge tape that was entered.	
REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the user request was processed.	
	Value	**Description**
	0	User request completed normally.
	1	Server error occurred for which there is no existing status.

AFTER-MOUNT NOTIFICATION EXIT

The AFTER-MOUNT notification exit passes data to a user program upon completion of a CSC-initiated cartridge tape mount. The notification exit data originates from OS 2200 at mount request time, and from the server at mount completion time.

CSC converts OS 2200 LOAD requests to library server mount requests. Each mount request sent to the server receives a response. CSC converts the information from the OS 2200 LOAD, and the server mount response, to AFTER-MOUNT format. CSC then passes the information to CSCUI. The user program can receive AFTER-MOUNT notifications by using either the ACCEPT or ACCEPTW control functions. These notification exits are passed through CSCUI exit number 4.

If CSC terminates after issuing a mount request, but before receiving a mount response, the mount is not reported.

NOTE

There are no AFTER-MOUNT notifications for cartridge mounts initiated from the server console of by other client systems.

0	☑EXIT-ID		**CSCUI-STATUS**	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2	**CTL-POOLNAME**			
3				
4				
5	☑VALID-FIELDS-PASSED			
6	**VALID-FIELDS-RETURNED**			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9		**ACS-ID**		**RETURND-EXIT-ID**
10	USER-SUPPLIED-REQUEST-ID			
11				
12	**EXTENDED-SCRATCH-POOL**		**SCRATCH-POOL**	WAIT-TIME
13	**VOLUME-ID**			
14			**TAPE-OPTIONS**	**REQ-COMPL-STAT**
15	**UNIT-NAME**			
16			**EXPIRATION-PERIOD**	
17	**REL-REEL**			
18	**QUALIFIER**			
19				
20				
21	**FILE-NAME**			
22				
23				
24	**FCYCLE**			
25	**ORIGINAL-RUN-ID**			
26			**GENERATED-RUN-ID**	
27				
28	**ACCOUNT-NO**			
29				
30				
31	**USER-ID**			
32				
33				

Figure 5-4. AFTER-MOUNT Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-4. AFTER-MOUNT Notification Exit Field Descriptions

☑

Field	Description	
EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	4	CSC notification exit(s) from which each AFTER-MOUNT data is accepted.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • invalid function • not enough fields specified • too many fields specified • wrong fields specified • no exits supplied
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSC DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated this function.
	13	User request timed out without receiving data from CSC.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
CSCUI-STATUS *(cont'd)*	17	The requesting program DEREGISTERed while the user request was waiting.
	21	Wrong version specified with new interface format.

Field	Description		
☑ NIF INDICATOR	99	New interface format indicator.	
☑ VERSION	1	Identifies the packet format version.	
☑ FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW	
CTL-POOLNAME	CTL-POOLNAME specified on the mount request.		
☑ VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000		
VALID-FIELDS-RETURNED	BASE: 000 464 177 066 OPTIONAL: 014 001 000 000		
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.		
ACS-ID	ACS in which the cartridge tape was mounted.		
RETURNED-EXIT-ID	4	AFTER-MOUNT	
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.		
EXTENDED-SCRATCH-POOL	Scratch pool returned from mount request.		
SCRATCH-POOL	This field is not present if the mount was not for a scratch cartridge tape. For scratch cartridge tapes, this is the CSC defined default pool number for the requested label type (i.e., labeled vs. unlabeled).		
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the user request waits indefinitely for a reply.		
VOLUME-ID	Cartridge tape that was mounted.		
TAPE-OPTIONS	A bitmapped field containing information about the mounted cartridge tape.		
	Bitmap	**Value**	**Description**
	000	0	Original OS 2200 assign statement was for a labeled scratch cartridge tape.
	001	1	Original OS 2200 assign statement was for an unlabeled scratch cartridge tape.
TAPE-OPTIONS *(cont'd)*	010	2	Write-protection was specified on the assignment of this cartridge tape.
	100	4	Original OS 2200 assign statement was for a labeled scratch tape and specified a CTL-POOLNAME.

<table>
<tr><th>Field</th><th colspan="3">Description</th></tr>
<tr><td></td><td>101</td><td>5</td><td>Original OS 2200 assign statement was for an unlabeled scratch tape and specified a CTL-POOLNAME.</td></tr>
<tr><td rowspan="3"><b>REQUEST-COMPLETION-STATUS</b></td><td colspan="3">Status returned from CSC or the server showing how the user request was processed.</td></tr>
<tr><td><b>Value</b></td><td colspan="2"><b>Description</b></td></tr>
<tr><td>0</td><td colspan="2">User request completed normally.</td></tr>
<tr><td><b>UNIT-NAME</b></td><td colspan="3">OS 2200 name of the unit on which the cartridge tape was mounted.</td></tr>
<tr><td><b>EXPIRATION-PERIOD</b></td><td colspan="3">Retention time, in days, specified on the cartridge tape assignment. If no retention time was specified, defaults to the OS 2200 configured default.</td></tr>
<tr><td><b>REL-REEL</b></td><td colspan="3">Relative reel number mounted. This is relative to the OS 2200 file.</td></tr>
<tr><td><b>QUALIFIER</b></td><td colspan="3">Qualifier for this cartridge tape file.</td></tr>
<tr><td><b>FILE-NAME</b></td><td colspan="3">File name specified on the OS 2200 @ASG statement for this cartridge tape file.</td></tr>
<tr><td><b>FCYCLE</b></td><td colspan="3">Absolute file cycle for this cartridge tape file. This is a character field.</td></tr>
<tr><td><b>ORIGINAL-RUN-ID</b></td><td colspan="3">Run ID specified on the @RUN statement for the run requesting this cartridge tape mount.</td></tr>
<tr><td><b>GENERATED-RUN-ID</b></td><td colspan="3">Run ID used by OS 2200 to uniquely identify the run that requested this cartridge tape.</td></tr>
<tr><td><b>ACCOUNT-NO</b></td><td colspan="3">OS 2200 account number of the run that requested this cartridge tape.</td></tr>
<tr><td><b>USER-ID</b></td><td colspan="3">OS 2200 user ID of the run that requested this cartridge tape.</td></tr>
</table>

BEFORE-MOUNT NOTIFICATION EXIT

The BEFORE-MOUNT notification exit passes data to a user program indicating that a mount has been requested by the OS 2200. The notification exit data originates from OS 2200 at LOAD or SERVICE request.

The user program can receive BEFORE-MOUNT notifications by using either the ACCEPT or ACCEPTW control functions. These notifications are passed through CSCUI exit number 0.

CSCUI returns BEFORE-MOUNT notifications only for MOUNT requests initiated by the OS 2200 on this client. Mounts initiated via the OS 2200 console, the server console, or from another client are not reported.

0	☑EXIT-ID		CSCUI-STATUS	☑NIF INDICATOR
1			☑VERSION	☑FUNCTION
2				
3	CTL-POOLNAME			
4				
5	☑VALID-FIELDS-PASSED			
6	VALID-FIELDS-RETURNED			
7	TIME STAMP (Word 0)			
8	TIME STAMP (Word 1)			
9		ACS-ID		RETURND-EXIT-ID
10	USER-SUPPLIED-REQUEST-ID			
11				
12	EXTENDED-SCRATCH-POOL		SCRATCH-POOL	WAIT-TIME
13	VOLUME-ID			
14			TAPE-OPTIONS	REQ-COMPL-STAT
15	UNIT-NAME			
16			EXPIRATION-PERIOD	
17	REL-REEL			
18				
19	QUALIFIER			
20				
21				
22	FILE-NAME			
23				
24	FCYCLE			
25	ORIGINAL-RUN-ID			
26			GENERATED-RUN-ID	
27				
28				
29	ACCOUNT-NO			
30				
31				
32	USER-ID			
33				

Figure 5-5. BEFORE-MOUNT Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-5. BEFORE-MOUNT Notification Exit Field Descriptions

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td rowspan="3">☑ EXIT-ID</td><td colspan="2">The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>CSC notification exit(s) from which each BEFORE-MOUNT data is accepted.</td></tr>
<tr><td rowspan="12"><b>CSCUI-STATUS</b></td><td colspan="2">Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></tr>
<tr><td>0</td><td>User request completed normally.</td></tr>
<tr><td>2</td><td>REQUEST-ID in the user request duplicates one from a currently active request.</td></tr>
<tr><td>4</td><td>User request not accepted, or not processed, because:
• invalid function
• not enough fields specified
• too many fields specified
• wrong fields specified
• no exits supplied</td></tr>
<tr><td>6</td><td>No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.</td></tr>
<tr><td>7</td><td>REGISTER control function was not performed.</td></tr>
<tr><td>9</td><td>Function is illegal for the requested notification exit(s).</td></tr>
<tr><td>10</td><td>CSC DEREGISTERed from CSCUI without returning data for the user request.</td></tr>
<tr><td>12</td><td>TERMINATE-REQUEST terminated this function.</td></tr>
<tr><td>13</td><td>User request timed out without receiving data from CSC.</td></tr>
<tr><td>14</td><td>Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.</td></tr>
</table>

Field	Description		
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.	
CSCUI-STATUS *(cont'd)*	17	The requesting program DEREGISTERed while the user request was waiting.	
	21	Wrong version specified with new interface format.	
☑ NIF INDICATOR	99	New interface format indicator.	
☑ VERSION	1	Identifies the packet format version.	
☑ FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW	
CTL-POOLNAME	CTL-POOLNAME specified on the mount request.		
☑ VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000		
VALID-FIELDS-RETURNED	BASE: 000 464 177 066 OPTIONAL: 014 001 000 000		
TIME STAMP	Time, in DWTIME$ format, when CSCUI processed each user request.		
ACS-ID	ACS in which the cartridge tape mount is requested.		
RETURNED-EXIT-ID	0	BEFORE-MOUNT	
USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time.		
EXTENDED-SCRATCH-POOL	Scratch pool from mount request.		
SCRATCH-POOL	This field is not present if the mount was not for a scratch cartridge tape. For scratch cartridge tapes, this is the CSC defined default pool number for the requested label type (i.e., labeled vs. unlabeled).		
WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the user request waits indefinitely for a reply.		
VOLUME-ID	Cartridge tape that is requested, or 0 if scratch tape requested.		
TAPE-OPTIONS	A bitmapped field containing information about the mount request.		
	Bitmap	**Value**	**Description**
	000	0	Original OS 2200 assign statement was for a labeled scratch cartridge tape.

Field	Description		
	001	1	Original OS 2200 assign statement was for an unlabeled scratch cartridge tape.
TAPE-OPTIONS *(cont'd)*	010	2	Write-protection was specified on the assignment of this cartridge tape.
	100	4	Original OS 2200 assign statement was for a labeled scratch tape and specified a CTL-POOLNAME.
	101	5	Original OS 2200 assign statement was for an unlabeled scratch tape and specified a CTL-POOLNAME.
REQUEST-COMPLETION-STATUS	Describes type of mount requested.		
	Value	**Description**	
	0	This BEFORE-MOUNT notification is the result of a LOAD request.	
	1	This BEFORE-MOUNT notification is the result of a SERVICE request.	
UNIT-NAME	OS 2200 name of the unit on which the cartridge tape is requested.		
EXPIRATION-PERIOD	Retention time, in days, specified on the cartridge tape assignment. If no retention time was specified, defaults to the OS 2200 configured default.		
REL-REEL	Relative reel number requested. This is relative to the OS 2200 file.		
QUALIFIER	Qualifier for this cartridge tape file.		
FILE-NAME	File name specified on the OS 2200 @ASG statement for this cartridge tape file.		
FCYCLE	Absolute file cycle for this cartridge tape file. This is a character field.		
ORIGINAL-RUN-ID	Run ID specified on the @RUN statement for the run requesting this cartridge tape mount.		
GENERATED-RUN-ID	Run ID used by OS 2200 to uniquely identify the run that requested this cartridge tape.		
ACCOUNT-NO	OS 2200 account number of the run that requested this cartridge tape.		
USER-ID	OS 2200 user ID of the run that requested this cartridge tape.		

EJECT-COMPLETION NOTIFICATION EXIT

The EJECT-COMPLETION notification exit passes the final status of an INITIATE-EJECT user request to the user program. See the discussion of the “INITIATE-EJECT User Request” on page 4-17 for a description of the entire user-initiated eject process.

CSC returns this notification after the eject has been completed. The RETURNED-REQUEST-ID associates EJECT-COMPLETION notification exits with INITIATE-EJECT user requests.

0	☑EXIT-ID	**CSCUI-STATUS**	☑NIF INDICATOR
1		☑VERSION	☑FUNCTION
2	n/a		
3	RESERVED		
4	RESERVED		
5	☑VALID-FIELDS-PASSED		
6	**VALID-FIELDS-RETURNED**		
7	**TIME STAMP (Word 0)**		
8	**TIME STAMP (Word 1)**		
9			**RETURND-EXIT-ID**
10	USER-SUPPLIED-REQUEST-ID		
11	**RETURNED-REQUEST-ID**		
12			WAIT-TIME
13			
14			**REQ-COMPL-STAT**

Figure 5-6. EJECT-COMPLETION Notification Exit Packet

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

Table 5-6. EJECT-COMPLETION Notification Exit Field Descriptions

☑

Field	Description	
EXIT-ID	The CSC notification exit(s) accessed for a specific user request. This field contains the numeric value of a single notification exit, or a bitmap of selected exits. Bit 17 of this field is set for the bitmap form and clear for the single value form.	
	Value	**Description**
	11	CSC notification exit(s) from which EJECT-COMPLETION data is accepted.
CSCUI-STATUS	Status returned from CSCUI for each user request, showing whether CSCUI accepted and processed the request.	
	Value	**Description**
	0	User request completed normally.
	2	REQUEST-ID in the user request duplicates one from a currently active request.
	4	User request not accepted, or not processed, because: • not enough fields specified • too many fields specified • wrong fields specified • an unused field does not contain zero
	6	No data was available at the indicated notification exit(s) when an ACCEPT control function was issued.
	7	REGISTER control function was not performed.
	9	Function is illegal for the requested notification exit(s).
	10	CSCAM DEREGISTERed from CSCUI without returning data for the user request.
	12	TERMINATE-REQUEST terminated the ACCEPTW control function.
	13	ACCEPTW control function timed out without receiving data from CSCAM.
	14	Data returned by the ACCEPT or ACCEPTW control function was truncated because the user request packet length specified by the user was not long enough.
	15	The user request packet is not completely contained within the main D-Bank, or the main D-Bank is write-protected.
CSCUI-STATUS *(cont'd)*	17	The requesting program DEREGISTERed while the user request was waiting.
	21	Wrong version specified with new interface format.

	Field	Description	
☑	NIF INDICATOR	99	New interface format indicator.
☑	VERSION	1	Identifies the packet format version.
☑	FUNCTION	1 or 2	1 = ACCEPT; 2 = ACCEPTW
☑	VALID-FIELDS-PASSED	REQUIRED: 000 000 000 000 OPTIONAL: 001 100 000 000	
	VALID-FIELDS-RETURNED	BASE: 000 000 240 000 OPTIONAL: 000 000 000 000	
	TIME STAMP	Time, in DWTIME$ format, when CSCUI processed this notification.	
	RETURNED-EXIT-ID	11	EJECT-COMPLETION
	USER-SUPPLIED-REQUEST-ID	Value supplied by the user program to identify each user request at a later time. This field is only valid for the ACCEPTW and TERMINATE-REQUEST control functions.	
	RETURNED-REQUEST-ID	Identifier, assigned by CSC, which the server uses to refer to a specific INITIATE-EJECT user request. The value of this field matches the RETURNED-REQUEST-ID of the original INITIATE-EJECT user request. This field is only present for eject notification exits generated by CSC at the completion of an INITIATE-EJECT user request.	
	WAIT-TIME	Number of seconds that each user request waits for a response. If this time elapses without receiving a response, a timeout CSCUI-STATUS 13 is returned. A value of 0 indicates that the user request waits indefinitely for a reply.	
	REQUEST-COMPLETION-STATUS	Status returned from CSC or the server showing how the request was processed.	
		Value	**Description**
		0	User request completed normally.
		1	Server error occurred for which there is no existing status.
	REQUEST-COMPLETION-STATUS *(cont'd)*	6	Specified LSM, ACS, or CAP is not in library.
		8	Indicates a parameter error for one of the following conditions: • Invalid CAP-COUNT • Invalid VOLUME-COUNT • Invalid characters in VOLUME-ID • Invalid ACS or LSM in CAP list

6. CSCUI LOGGING

This chapter describes CSCUI logs and logging functions. CSCUI logging allows a user program to record significant events occurring in CSCUI, including these functions:

- Crucial cartridge tape management information that CSCUI passes to user programs.

- User requests, which maintain synchronization between the TLMS and the server.

OVERVIEW

During the CSC build, you can configure CSCUI to log its requests in the OS 2200 system log.

CSCUI writes user program activities into the OS 2200 system log in a user log record format. The size of these records is small (less than 140 words), so there is little impact on the system log, even with high mount activity.

Users can develop programs to read the system log file and extract user records logged by CSCUI. This may be an important source for recovering data lost during operational problems with a user program registered with CSCUI.

WARNING

If you are still using CSCUI programs created before 2R5, the CSCUI log entries may contain mixed-format records.

You should check with the staff responsible for building and installing CSC to determine the OS 2200 system log user record type chosen for these log records. The default is 50302.

CSCUI LOGGING FUNCTIONS

CSCUI can create entries in the system log for requests sent by user programs and by CSC. This provides a history of interactions with CSC. Different log subtypes are used for requests that were successfully passed through CSCUI and for those where the pass did not complete. The entries for incomplete interactions may be used by programs that normally receive information from CSCUI to recover missed notifications.

You should coordinate your needs for CSCUI log records with your site administrator. During the configuration of CSCUI, your site administrator can define the following:

- the entry type for log records produced by CSCUI,
- which CSCUI requests and notifications will produce entries in the system log,
- which transfer direction is logged(to CSC, from CSC, or both).

CSCUI Log Record Format

The CSCUI-generated log entry contains a log record descriptor and a copy of the CSCUI request packet. The request packet contains either a user request, a CSC responses to a user request, or a notification from CSC. The log record length varies with the size of the user packet.

This log record subtype tells why CSCUI produced the log record. Table 6-1 shows the log record subtypes and their meaning.

Table 6-1. CSCUI Log Record Subtypes

Value	Log Record Content
1	Packet accepted by CSCUI that is either a request from a user program or a CSC notification
2	Response packet to a user request accepted by CSCUI from CSC
3	A valid user request or CSC notification packet rejected by CSCUI because that exit queue was full
4	A CSC notification packet that was in the queue when a registration was done that specified the queue flush option
5	A reply packet from CSC when the requesting user cannot be found
6	A packet purged from the bank through TRMRG$ contingency processing
64	The CSC REGISTER packet
65	The CSC DEREGISTER packet

Although the different CSCUI request packets contain varied data items, the first eight (8) words of the packet are identical. The following figure (6-1) shows the format of these common fields and highlights those that are useful for identifying the log record content.

0	**EXIT-ID**		CSCUI-STATUS	NIF INDICATOR
1			VERSION	**FUNCTION**
2 3 4	CTL-POOLNAME			
5	VALID-FIELDS-PASSED			
6	VALID-FIELDS-RETURNED			
7	**TIME STAMP (Word 0)**			
8	**TIME STAMP (Word 1)**			
9	VOLUME-COUNT	ACS-ID	LSM-ID	RETURND-EXIT-ID
10	**USER-SUPPLIED-REQUEST-ID**			
11 : n	*Remainder of the user request or notification packet. Contents of these fields is determined by the exit-id value. See Table 6-2 below*			

Figure 6-1. CSCUI Log Record Format

Table Legend

Bold text	returned parameter
Normal text	passed parameter
☑	required field

The table below (6-2) describes the fields in the log record that are relevant to identifying log records.

Table 6-2. CSCUI Log Record Field Descriptions

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td><b>EXIT-ID</b></td><td colspan="2">This identifies the type of packet in this log entry. Appendix A contains definitions of the possible values.</td></tr>
<tr><td rowspan="5"><b>FUNCTION</b></td><td colspan="2">This tells what function the sender of this packet asked of CSCUI. The function is one of the following:</td></tr>
<tr><td><b>Value</b></td><td><b>Description</b></td></tr>
<tr><td>9</td><td>User request being sent to CSC</td></tr>
<tr><td>11</td><td>A notification being sent from CSC</td></tr>
<tr><td>12</td><td>A response from CSC to a user request</td></tr>
<tr><td><b>TIME STAMP</b></td><td colspan="2">This is the time in DW$TIME format when CSCUI received this request packet.</td></tr>
<tr><td><b>USER-SUPPLIED-REQUEST-ID</b></td><td colspan="2">In a response from CSC to a user request, this field will contain the same value as the original user request.</td></tr>
</table>

What log records are generated

Assuming that your site has configured CSCUI to produce all possible log records, a typical user request will produce the following log records:

- A record with subtype 1 and function 9 when the request is passed to CSCUI
- A record with subtype 2, function 12, and the same USER-SUPPLIED-REQUEST-ID when CSC returns a response

A typical notification from CSC will produce the following log records:

- A record with subtype 1 and function 11 when CSC passes the notification to CSCUI
- A record with subtype 3 and function 11 when the user program receives the notification

Log records with subtype 4, 5, or 6 are produced when a packet is not successfully passed through CSCUI between CSC and a user program. The function tells why the packet was passed to CSCUI.

APPENDIX A. REFERENCE TABLES

This appendix contains tables for quick reference:

- a bit-mapped table showing values for the VALID-FIELDS-PASSED field
- a complete list of CSCUI-STATUS codes
- a complete list of CSCUI Exit IDs

VALID-FIELDS-PASSED FIELD VALUES

Table A-1 is a bit-mapped table showing the VALID FIELDS PASSED field definition. Whenever a CSCUI-STATUS value of 4 is returned, two bitmaps will be displayed, VALID-FIELDS-PASSED and VALID-FIELDS-EXPECTED. Use the following table to “OR” the fields you want to pass to CSCUI, then compare the results to the VALID-FIELDS-EXPECTED string. They should match. Note that CSCUI does *not* display optional fields.

Table A-1. VALID-FIELDS-PASSED Field Definitions

Field	Description
000 000 000 000 000 000 000 000 000 000 000 001	not used
000 000 000 000 000 000 000 000 000 000 000 010	ACCOUNT-NUMBER
000 000 000 000 000 000 000 000 000 000 000 100	ACS-ID
000 000 000 000 000 000 000 000 000 000 001 000	not used
000 000 000 000 000 000 000 000 000 000 010 000	EXPIRATION-PERIOD
000 000 000 000 000 000 000 000 000 000 100 000	GENERATED RUN-ID
000 000 000 000 000 000 000 000 000 001 000 000	LAST-SELECT-TIME
000 000 000 000 000 000 000 000 000 010 000 000	LSM-ID
000 000 000 000 000 000 000 000 000 100 000 000	OPTIONS
000 000 000 000 000 000 000 000 001 000 000 000	ORIGINAL-RUNID
000 000 000 000 000 000 000 000 010 000 000 000	QUALIFIER
000 000 000 000 000 000 000 000 100 000 000 000	FILE-NAME
000 000 000 000 000 000 000 001 000 000 000 000	FCYCLE
000 000 000 000 000 000 000 010 000 000 000 000	REL-REEL
000 000 000 000 000 000 000 100 000 000 000 000	REQUEST-COMPLETION-STATUS
000 000 000 000 000 000 001 000 000 000 000 000	RETURNED EXIT-ID
000 000 000 000 000 000 010 000 000 000 000 000	RETURNED REQUEST-ID and USER-SUPPLIED-REQUEST-ID and RETURNED-SERVER-STATUS
000 000 000 000 000 000 100 000 000 000 000 000	SCRATCH-INDICATOR
000 000 000 000 000 001 000 000 000 000 000 000	SCRATCH-POOL and SCRATCH-POOL-INDICATOR
000 000 000 000 000 010 000 000 000 000 000 000	SELECT-COUNT
000 000 000 000 000 100 000 000 000 000 000 000	TAPE-OPTIONS
000 000 000 000 001 000 000 000 000 000 000 000	TIME-INSERTED
000 000 000 000 010 000 000 000 000 000 000 000	UNIT-NAME
000 000 000 000 100 000 000 000 000 000 000 000	USER-ID
000 000 000 001 000 000 000 000 000 000 000 000	USER-SUPPLIED-REQUEST-ID
000 000 000 010 000 000 000 000 000 000 000 000	VOLUME-COUNT
000 000 000 100 000 000 000 000 000 000 000 000	VOLUME-ID
000 000 001 000 000 000 000 000 000 000 000 000	WAIT-TIME
000 000 010 000 000 000 000 000 000 000 000 000	USER-LSM and VOLRPT-ID
000 000 100 000 000 000 000 000 000 000 000 000	CTL-POOLNAME
000 001 000 000 000 000 000 000 000 000 000 000	EXTENDED-ACS-POOL-NBR
000 010 000 000 000 000 000 000 000 000 000 000	LEVEL-INFO
000 100 000 000 000 000 000 000 000 000 000 000	CAP-NUMBER

CSCUI-STATUS CODES

Table A-2 lists all of the possible CSCUI-STATUS codes.

Table A-2. CSCUI-STATUS Codes

Code	Description
0	User request processed normally.
1	Notification exit queue full.
2	Duplicate user request.
3	Invalid notification exit id.
4	Invalid packet contents.
5	Invalid user request id.
6	No data available for ACCEPT control function.
7	User not registered.
8	Maximum number of users registered.
9	User not authorized.
10	Peer deregistered without ACCEPT or SEND control function.
11	Peer not registered.
12	User request terminated via TERMINATE-REQUEST control function.
13	Timeout has occurred.
14	Truncation of ACCEPT control function data due to packet size.
15	Invalid packet address.
16	No waiting user request found.
17	Program has DEREGISTERED.
18	User has requested a queue flush.
19	Timing routine terminated.
20	Common bank reload/configure.
21	Wrong version specified with New Interface Format.

CSCUI EXIT IDS

Table A-3 lists all of the possible CSCUI Exit IDs.

Table A-3. CSCUI Exit IDs

Exit ID	Description
0	BEFORE-MOUNT
1	MOUNT
2	AFTER-EJECT
3	AFTER-ENTER
4	AFTER-MOUNT
5	AFTER-DISMOUNT
6	DO-VOLRPT
7	DO-ENTER
8	INITIATE-EJECT
9	SCRATCH/UNSCRATCH
10	VOLUME-INFORMATION
11	EJECT-COMPLETION
12	INITIATE-VOLRPT*
13	VOLRPT-STATUS*
14	CLEAN*
15	UNIT-LIST*
16	SCRATCH-LIST*

* VM server only

NOTE

Several CSCUI Exit IDs listed in Table A-3 apply to CSC series 2RX and the VM server only. They are included here for completeness.

APPENDIX B. SAMPLE PROGRAMS

The basic flow of a program that uses CSCUI consists of a REGISTER function followed by any number of CSCUI requests and/or notifications. When all requests are completed, the program uses the DEREGISTER function to disconnect from CSCUI. Each of the CSCUI requests and notifications are described in detail in the pages of this manual. This appendix describes sample programs that are included with CSC to demonstrate how to build applications that use CSCUI.

During the CSC installation, a file is created containing programming interface elements and sample programs. The name of this file is the same as the CSC installation file with "-2" appended to the file name. The default is SYS$LIB$*CSC-2. In this appendix, this will be referred to as the CSCUI file.

The CSCUI file contains the following types of elements:

- CSCUI packet definitions
- CSCUI interface functions in source and compiled versions
- CSCUI definition elements for use in linking or mapping your program
- Sample programs in various languages

CSCUI PACKET DEFINITIONS

The elements USERCSCUIDEF and CSCUI/H in the CSCUI file contain the MASM and C definitions for CSCUI packets respectively. There is no element dedicated to COBOL packet definitions. The COBOL sample programs contain CSCUI packets definitions within the WORKING STORAGE SECTION. You can use these as templates for coding packet definitions. The packet layout for a given function must follow the data definitions published in this manual.

The following example is a skeleton to indicate how to construct the COBOL data definition.

```
01  PACKET-ADDR.
   03  field-1              PIC X(xx).
    03  field-2              PIC X(xx).
         .
         .
    03  field-n              PIC X(xx).
01  PACKET-LEN               PIC 9(10) COMP.
01  STATUS                   PIC 9(10) COMP.
```

All parameters must be on word boundaries (77/01 level). You should code fields with binary values as usage COMPUTATIONAL (COMP).

CSCUI INTERFACE FUNCTIONS

The ACOBCSCUI and UCS$CSCUI elements contain the code to interface between your program and CSCUI. ACOBCSCUI is for ASCII COBOL programs and UCS$CSCUI is for the UCS compiler programs. The source code and compiled versions of each interface element are provided.

ACOBCSCUI

The element ACOBCSCUI contains the function that interfaces CSCUI to an ASCII COBOL program. A$CSCUI is the entry point into this function. Use the following call to invoke a CSCUI function from an ASCII COBOL program:

```
CALL 'A$CSCUI' USING PACKET-LEN, PACKET-ADDR, STATUS.
```

If the COBOL program does not pass the appropriate parameters to CSCUI, the program is aborted and the following error message displays:

```
*Error invalid number of parameters passed to A$CSCUI.
```

UCS$CSCUI

This element UCS$CSCUI contains the function that interfaces CSCUI to UCS programs. This interface, written in MASM, is callable by both C and COBOL. Although there is only one function, four calling sequences are provided for C programs and two for COBOL programs. The calling sequences differ in the number of parameters and the type of parameter that is used for the CSCUI packet length.

UCS C

The C prototypes for the CSCUI functions are provided in the CSCUI/H element. The four prototypes allow C to detect errors in the type or number of parameters passed.

The C prototypes are:

1. `int UCS$CSCUI1(int p_pkt_sz, char *p_pkt);`
2. `int UCS$CSCUI2(int *p_pkt_sz, char *p_pkt);`
3. `int UCS$CSCUI3(int *p_pkt_sz, char *p_pkt, int *p_stat);`
4.
```
int UCS$CSCUI4(int *p_pkt_sz, char *p_pkt, int *p_stat,
               int *exits);
```

where:

`p_pkt_sz` is the size of the CSCUI packet in words, passed and returned.
`p_pkt` is the CSCUI packet.
`p_stat` is the CSCUI status that is returned from the CSCUI call.
`exits` is a bit-mapped value containing the configured exits.

In addition to the four prototypes, CSCUI/H has provisions to `#define` UCS$CSCUI to any one of the four prototypes. This is requested by a `#define INCL_CSCUI_FORMAT_`*`n`* before the `#include CSCUI/H`. The value of n determines which prototype is associated with UCS$CSCUI. The default is prototype 2.

If no definition for UCS$CSCUI is wanted, the user can #DEFINE NO_INCL_CSCUI_FORMAT in their program. If CSCUI/H is not included, then UCS$CSCUI and the prototypes are not defined.

The C calling sequences are as follows:

1. `status = UCS$CSCUI1(pkt_sz, &pkt);`

2. `status = UCS$CSCUI2(&pkt_sz, &pkt);`
3. `status = UCS$CSCUI3(&pkt_sz, &pkt, &stat);`
4. `status = UCS$CSCUI4(&pkt_sz, &pkt, &stat, &exits);`

where:

`status` is the status returned by CSCUI or one of the following status values if UCS$CSCUI detected an error before calling CSCUI:

-1 invalid number of parameters
-2 invalid parameter format or size
-3 invalid CSCUI BDI configured

`pkt_sz` is the size of the CSCUI packet in words, passed and returned.

`pkt` is the CSCUI packet.

`stat` is the CSCUI status that is returned from the CSCUI call.

`exits` is the bit-mapped value containing the configured exits.

UCS COBOL

UCS$CSCUI can be called from COBOL using either three or four parameters. The calls are:

1. `CALL 'UCS$CSCUI' USING pkt-len, pkt, stat.`
2. `CALL 'UCS$CSCUI' USING pkt-len, pkt, stat, exits.`

where:

`pkt-len` is the PIC 9(10) binary length in words.

`pkt` is the CSCUI packet definition.

`stat` is the PIC 9(10) binary CSCUI status or one of the following values if UCS$CSCUI detected an error before calling CSCUI:

-1 invalid number of parameters
-2 invalid parameter format or size
-3 invalid CSCUI BDI configured

`exits` is the PIC 1(36) binary bit-mapped value of the configured exits.

LINKing and MAPping

When a program that uses CSCUI is LINKed or MAPped, the CSCUI common bank definition element must be included. For all but MASM programs, the element containing the common bank interface function must also be included.

The common bank definition elements are CBEP$CSC and CBEP$$CSC. They contain BDI and entry point information for calling CSCUI. CBEP$CSC is

LINKED with extended mode programs and CBEP$$CSC is MAPped with basic mode programs.

The interface functions are in the elements ACOBCSCUI and UCS$CSCUI. ACOB programs must include ACOBCSCUI during the MAP. UCS programs must include UCS$CSCUI during the LINK.

Assuming that the CSCUI file has a use name of CSC-2, the following directives would be included in the MAP of an ACOB program using CSCUI:

```
IN CSC-2.ACOBCSCUI, CSC-2.CBEP$$CSC
```

Similarly, the following directive would be included in the LINK of a UCS C or COBOL program.

```
IN CSC-2.UCS$CSCUI, CSC-2.CBEP$CSC
```

Snaps of the CSCUI packets can be produced in the UCS$CSCUI interface to aid in debugging your program. The snaps are enabled by re-LINKing your program with the following LINK directive:

```
CHANGE REFERENCE (CSCUIDBGOFF) TO CSCUIDBGON
```

NOTE

The extended mode interface is not designed for simultaneous use by multiple activities. If you have a multi-activity program, only one activity can use this interface.

SAMPLE PROGRAMS

The CSCUI file contains sample programs to illustrate how the various CSCUI requests are combined to produce a program. Each sample program has an element name with the following format: LANGsample_id/TYPE. LANG is the language and is one of ACOB, MASM, UC, or UCOB. Sample_id is usually a descriptive name for the CSCUI function demonstrated by the program. TYPE is "SOURCE" for the program source, "COMPILE" for the ECL to compile the sample, or "EXECUTE" for the ECL to execute the sample program.

The following table (B-1) describes the sample programs provided with CSC. The LANG portion of the program name is underlined. The "Func" column tells the functional status the sample program using one of the following values.

No -- The program cannot be used without modification.

Lim -- The program is functional but has limitations due to hard coded data values.

Yes -- The program can be used without modification.

Table B-1. Sample Programs Provided with CSC

Sample Name	Function	Program Description
ACOBAFTMOUNT	Yes	Receives one after mount notification and displays the volume and the unit on which it was mounted.
ACOBUNSCRTCH	Lim	Issues a set scratch request for one hard coded volume id.
ACOBUNSCR42	Lim	Issues a set scratch request containing 42 hard coded volume ids.
MASMSAMPLE	No	Repeatedly does the following: - Wait for enter notification - Execute user supplied code to determine if the entered volume should be a scratch tape. - If the volume should be scratched, then issue a set scratch request
UCDOEJECT	Yes	Uses information from the processor call to issue an eject request for up to 42 volumes. Based on processor call options, this program may also create an activity to receive the after eject notifications.
UCDOVOLRPT	Yes	Uses information from the processor call to issue a do volume report request.
UCOBVOLINFO	Lim	Issues a volume information request for a hard coded volume id. There is no program output if the volume info request completes normally.
UCSCRATCH	Lim	Reads input and issues one set scratch request for the volume id in each input image. This program always scratches to pool 5.
UCSCRTCH42	Lim	Reads input containing volumes to be scratched. Up to 42 volumes are batched together and sent in one set scratch request. This program always scratches to pool 5.

INDEX

A

ACODCSCUI
- interface to CSCUI, B-2

ACS
- ejecting tapes, 2-6
- entering tapes, 2-6
- physically removing tapes from, 2-8

ACS library software, 2-2

ACS statement, 4-7

Addressing
- restrictions for CSCUI control functions, 3-5

AFTER-DISMOUNT notification exit, 5-2

AFTER-EJECT notification exit, 5-7

AFTER-ENTER notification exit, 5-11

AFTER-MOUNT notification exit, 5-14

Automated Cartridge System. *see* ACS

B

Batch ejects, 2-7

BEFORE-MOUNT notification exit, 5-19

C

Calling
- CSCUI control functions, 3-2
- MASM interface, 3-2
- notification exits, 5-2
- UCOB interface, 3-3
- UCS C interface, 3-3
- user requests, 4-2

Client System Component User Interface. *see* CSCUI

Command syntax notation, xiii

Common bank
- addresses, 3-5
- reloading, 3-4

Control functions
- addressing, 3-5
- calling, 3-2
- CSCUI, 3-1
- registers used by, 3-2
- restrictions, 3-4

CSCUI
- control function descriptions, 3-6
- control functions, 3-1
  - addressing, 3-5
  - calling, 3-2
  - DEREGISTER, 3-9
  - REGISTER, 3-6
  - registers, 3-2
  - restrictions, 3-4
  - TERMINATE-REQUEST, 3-11
- exit ids, reference table, A-4
- interface functions, B-2
  - ACOBCSCUI, B-2
  - UCS C, B-3
  - UCS COBOL, B-4
  - UCS$CSCUI, B-3
- introduction, 1-1
- linking and mapping programs, B-4
- log record format, 6-3
- logging, 6-1
- logging functions, 6-2
- logging, overview of, 6-2
- notification exits, 5-1
- overview, 1-2
- sample programs, B-1
- services
  - control functions and, 3-4

CSCUI (*cont'd*)
  status codes
    reference table, A-3
  user requests, 4-1
  using, 3-5

D

DEREGISTER
  CSCUI control function, 3-9
Documentation
  usage guidelines, xi
DO-ENTER user request, 4-2
DO-VOLRPT user request, 4-6

E

EJECT command
  multiple requests, 2-8
  using, 2-8
EJECT-COMPLETION notification exit, 5-23
Ejecting cartridge tapes, 2-6
  options, 2-7
Ejects
  batch, 2-7
  manual, 2-7
  multiple concurrent, 2-9
  on-line, 2-7
ENTER command
  using, 2-6
Exit IDs
  CSCUI reference table, A-4
Exits. *see* Notification exits

F

FORMAT statement, 4-8
Functions
  server, 2-2
  TLMS, 2-2

G

Generating log records, 6-5

H

HEADINGS statement, 4-10

I

INITIATE-EJECT user request, 4-16
Input, 4-12
Interface functions
  CSCUI, B-2
Introduction to CSCUI, 1-1

L

Linking programs to CSCUI, B-4
LIST statement, 4-10
Log records
  CSCUI and, 6-2
  generation, 6-5
  record format, 6-3
  subtypes, 6-3
Logging
  CSCUI, 6-1
  overview, 6-2
Logging functions
  CSCUI, 6-2
LSM statement, 4-11

M

Manual ejects, 2-7
Mapping programs to CSCUI, B-4
MASM calling interface, 3-2
MOUNT user request, 4-22
Moving cartridge tapes, 2-6
Multiple eject requests
  CSCUI and, 2-8

N

Notification exits
  AFTER-DISMOUNT, 5-2
  AFTER-EJECT, 5-7
  AFTER-ENTER, 5-11
  AFTER-MOUNT, 5-14
  BEFORE-MOUNT, 5-19
  calling, 5-2
  CSCUI, 5-1
  EJECT-COMPLETION, 5-23

O

On-line ejects, 2-7
OUTPUT FILE statement, 4-11
Overview
  CSCUI, 1-2

P

Packet definitions
  sample program, B-2
Packet formats
  CSCUI, 3-6
Parameters
  volume report, 4-6
Programs
  samples using CSCUI, B-1

R

REGISTER
  CSCUI control function, 3-6
Registers
  CSCUI control functions and, 3-2
Reloading common banks, 3-4
Requests. *see* User requests

S

Sample programs, B-1
  CSCUI packet definitions, B-2
Sample programs in the CSCUI file, B-5
Scratch pools, 2-5
Scratch synchoronization
  maintaining, 2-5
Scratch tapes
  ACS view of tape status, 2-4
  ACS-TLMS interactions, 2-4
  client view of tape status, 2-3
  maintaining synchronization, 2-5
  TLMS and, 2-3
SCRATCH/UNSCRATCH user request, 4-29
Server
  TLMS interactions, 2-3
Server functions, 2-2
SORT statement, 4-12
Status codes
  CSCUI-STATUS reference table, A-3
Syntax
  notation guidelines, xiii

T

Tape Library Management System. *see* TLMS
Tape movement
  tracking with CSCUI, 2-3
Tapes
  batch ejects, 2-7
  ejecting from ACS, 2-6
  entering into ACS, 2-6
  inventory of cartridge tapes, 2-3
  manual ejects, 2-7
  movement of, 2-6
  on-line ejects, 2-7
  ownership and TLMS, 2-3
  physically removing from ACS, 2-8
  scratch pools, 2-5
  scratch status, 2-3
  scratching and TLMS, 2-3
  TLMS and inventory, 2-3
TERMINATE-REQUEST
  CSCUI control function, 3-11
TLMS, 2-1
  functions of, 2-2
  interactions with server, 2-3
  scratch interactions with ACS, 2-4
  scratch tapes and, 2-3
  tape inventory and, 2-3
  tape ownership and, 2-3

U

UCOB calling interface, 3-3
UCS C
  CSCUI functions and, B-3
UCS C calling interface, 3-3
UCS COBOL
  CSCUI functions and, B-4
UCS$CSCUI
  interface to CSCUI, B-3
User requests
  calling, 4-2
  CSCUI, 4-1
  DO-ENTER, 4-2
  DO-VOLRPT, 4-6
  INITIATE-EJECT, 4-16
  MOUNT, 4-22
  SCRATCH/UNSCRATCH, 4-29
  VOLUME-INFORMATION, 4-34
Using CSCUI, 3-5

V

VALID-FIELDS-PASSED field
  control functions and, 3-4
  reference table, A-2
VALID-FIELDS-RETURNED field
  control functions and, 3-4
Volume report parameters, 4-6
VOLUME-INFORMATION user request, 4-34

W

WAIT-TIME field
  control functions and, 3-4

Z

ZERO statement, 4-12

EFFECTIVE PAGES

Manual:	CSCUI Programmer’s Reference Manual 312537401
Issue Date:	February 2005
Reissue Date:	

This document has 124 total pages, including:

Title
Copyrights, Disclaimers & Trademarks
Contents, Tables & Figures

Preface	xi	thru	xiv
Chapter 1:	1-1	thru	1-2
Chapter 2:	2-1	thru	2-10
Chapter 3:	3-1	thru	3-12
Chapter 4:	4-1	thru	4-38
Chapter 5:	5-1	thru	5-26
Chapter 6:	6-1	thru	6-6
Appendix A:	A-1	thru	A-4
Appendix B:	B-1	thru	B-6
Index	Index-1	thru	Index-4

Effective Pages
Reader’s Comment Form
Business Reply Mailer