UNISYS

# A Series
# Communications Management System (COMS)
# **Migration Guide**

Release 3.9.0

September 1991

Priced Item

Printed in U S America
8600 1567–000

UNISYS

# A Series Communications Management System (COMS)

# Migration Guide



Release 3.9.0

September 1991

Printed in U S America

Priced Item

8600 1567–000

The names, places, and/or events used in this publication are not intended to correspond to any individual, group, or association existing, living, or otherwise. Any similarity or likeness of the names, places, and/or events with the names of any individual, living or otherwise, or that of any group or association is purely coincidental and unintentional.

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication may be forwarded using the Product Information card at the back of the manual, or may be addressed directly to Unisys, Product Information, 25725 Jeronimo Road, Mission Viejo, CA 92691.

# Page Status

| Page | Issue |
|---|---|
| iii | –000 |
| iv | Blank |
| v through vii | –000 |
| viii | Blank |
| ix through xi | –000 |
| xii | Blank |
| xiii | –000 |
| xiv | Blank |
| xv | –000 |
| xvi | Blank |
| 1–1 through 1–6 | –000 |
| 2–1 through 2–13 | –000 |
| 2–14 | Blank |
| 3–1 through 3–10 | –000 |
| 4–1 through 4–5 | –000 |
| 4–6 | Blank |
| 5–1 through 5–6 | –000 |
| A–1 through A–8 | –000 |
| B–1 through B–3 | –000 |
| B–4 | Blank |
| Glossary–1 through 9 | –000 |
| Glossary–10 | Blank |
| Bibliography–1 | –000 |
| Bibliography–2 | Blank |
| Index–1 through 4 | –000 |

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–*xyz*) indicates the document level. The first digit of the suffix (*x*) designates a revision level; the second digit (*y*) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (*z*) is used to indicate an errata for a particular level and is not reflected in the page status summary.

# About This Guide

## Purpose

This guide explains how to migrate from existing message control systems (MCSs) to the current release of the Communications Management System (COMS). COMS is an MCS designed for use with Unisys A Series systems. The COMS product is a member of the InterPro™ (Interactive Productivity) family of products.

## Scope

The guide details the strategies, tools, and methods of evaluation available for migration to COMS.

This guide does not include information on migrating from one release to another. For information on compatibility issues across releases, refer to the Software Release Capabilities Overview for the appropriate Mark release level. For example, the *A Series Mark 3.9 Software Release Capabilities Overview* includes information about the Mark 3.9 compatibility issues.

## Audience

The guide is written for the system administrators and system programmers who are responsible for migration issues at their site.

## Prerequisites

The users of the guide must be familiar with each existing MCS at their site and with COMS.

---

InterPro is a trademark of Unisys Corporation.

# How to Use This Guide

This guide is divided into five sections. Read section 1 when you are planning or reviewing your strategy for migrating to COMS. Section 2 is important if you intend to operate a user-created or Unisys MCS under COMS.

Sections 3 and 4 detail different aspects of migrating from the Generalized Message Control System (GEMCOS) to COMS, and should be read if you are performing this migration.

Section 5 gives you information on how remote files are handled if you migrate to COMS, and compares this handling with that of the Command and Edit language (CANDE) system.

For convenience, the symbols that appear in this guide are used in the same manner as they are used in related documents. For example, square brackets ( [ ] ) are used as part of GEMCOS declarations, and angle brackets ( < > ) denote metatokens and variable information.

Since the guide is addressed to users who are familiar with existing MCSs at their site and with COMS, many of the terms, acronyms, and concepts in the guide that are integral to COMS, GEMCOS, or other Unisys MCSs are not included in the glossary or index.

Unless otherwise stated, all documents referenced in the guide are A Series documents.

# Organization

The individual sections are described in the following outline of the document. In addition, a glossary, a bibliography, and an index appear at the end of this guide.

### Section 1. Migrating to COMS

This section outlines the major benefits of using COMS, the migration strategies, and the factors to consider when selecting a migration path for your site.

### Section 2. MCS Windows

This section describes the MCS window feature of COMS and provides information to help system programmers decide if any changes are needed to make a given MCS operate in a COMS environment.

### Section 3. Format Support Library

One tool used in the direct migration from GEMCOS to COMS, the format support library processing item, is examined. The section covers the preparation, installation, and use of the format support library to retain your GEMCOS input formatting, output formatting, form requests, and paging.

**Section 4. GEMCOS Windows**

Guidelines are presented for running GEMCOS in an MCS window, and various aspects of GEMCOS operation through COMS are considered.

**Section 5. Remote Files**

This section compares the handling of remote files through COMS with the handling of remote files through CANDE. Emphasis is placed on how COMS handles remote files.

**Appendix A. Comparing GEMCOS to COMS**

This appendix compares GEMCOS and COMS functions and features.

**Appendix B. GEMCOS Sources for COMS Entities and Attributes**

This appendix gives the GEMCOS sources for COMS entities and attributes.

# Related Product Information

For more information about COMS, refer to the following documents:

***A Series Communications Management System (COMS) Capabilities Manual* (form 8600 0627)**

This manual introduces COMS, discusses the flexibility and efficiency of the COMS system, describes the COMS architecture, and discusses specific features available to the COMS user. This manual is written for upper management, the site manager, and the programming staff.

***A Series Communications Management System (COMS) Configuration Guide* (form 8600 0312)**

This guide provides an overview of the basic concepts and functions of COMS. It includes instructions for creating a working COMS configuration and information on how to monitor and fine-tune COMS system performance. This guide is written for installation analysts, system analysts, programmers, administrators, and performance analysts.

***A Series Communications Management System (COMS) Operations Guide* (form 8600 0833)**

This guide explains how to perform terminal-based COMS functional tasks and serves as a reference to COMS commands. Syntax diagrams of COMS commands are provided with explanations and examples of how the commands can be used. This guide is written for terminal operators and computer operators.

***A Series Communications Management System (COMS) Programming Guide* (form 8600 0650)**

This guide explains how to write online, interactive, and batch application programs that run under COMS. This guide is written for experienced applications programmers with knowledge of data communication subsystems.

# Contents

# Figures

# Tables

# Section 1
# Migrating to COMS

This section outlines the benefits of the full-featured Communications Management System (COMS), the available routes and tools for migration to COMS, and the factors that influence migration.

## Why COMS?

COMS is a part of the Unisys InterPro (Interactive Productivity) family of products. It serves as the message control system (MCS) and supports processing on the Unisys A Series systems.

COMS allows you to select only the features you need for your processing. You can choose features such as a menu-driven user interface provided by the Menu-Assisted Resource Control (MARC), transaction processing, and real-time configuration changes.

COMS windows enable you to operate a number of program environments simultaneously at a given station. Without logging off, you can switch from window to window and retain the states of your window dialogs.

The close integration of COMS with the operating system guarantees high performance, especially for the direct window interface.

The available COMS processing items and windows are important factors in your migration plan.

### COMS Versions

Two versions of COMS are available to you, and both use the COMS configuration file to provide the message control environment. The full-featured version of COMS enables you to make changes to the configuration file. The COMS (Kernel) version comes with a predefined configuration file that cannot be manipulated.

#### Full-Featured COMS

The full-featured version of COMS offers the following:

- Eight dialogs each on the Menu-Assisted Resource Control (MARC) window, the Command and Edit (CANDE) window, and the Generalized Message Control System (GEMCOS) window for each connection.
- Preprocessing and postprocessing of messages.
- Routing by transaction code (trancode).

- User-defined usercodes and station restrictions.
- Dynamic control of user-created interface programs.
- Security options, such as security by trancode.
- Batch processing.
- Synchronized recovery.
- GEMCOS-to-COMS migration aids.

### COMS (Kernel)

COMS (Kernel) is a version of COMS that has minimum features compared to the full-featured version of COMS. COMS (Kernel) comes with a predefined configuration file, but does not include the reconfiguration capabilities found in the COMS Utility program. The COMS (Kernel) configuration file enables you to use the window feature, but does not include the COMS Utility window. Although COMS (Kernel) is mentioned here, some of the features discussed in this guide might be found only in the full-featured version of COMS.

## COMS Windows and Dialogs

COMS windows and dialogs are basic to most migration strategies. Four windows are defined by COMS at system initialization:

- MARC window
- COMS Utility window
- CANDE window
- GEMCOS window

The MARC window handles network control, data comm errors, and COMS termination. This window is a menu-driven interface to system resources that can be selected for an installation. The MARC window is similar to CANDE in the way it runs and controls application programs.

The COMS Utility window is accessible only from a control station, from the operator display terminal (ODT), or through a control-capable usercode. Through the COMS Utility window, you can define and maintain the COMS control file.

The CANDE and GEMCOS windows provide access to CANDE and GEMCOS, respectively.

There are three types of COMS windows:

- Direct windows
- MCS windows
- Remote-file windows

Direct windows are used to perform transaction-oriented processing that provides database synchronized recovery. MCS windows access message control systems other than COMS. Remote-file windows access remote-file programs.

The MARC and COMS Utility windows are direct windows. The CANDE and GEMCOS windows are MCS windows. Both direct and MCS windows can be declared through the COMS Utility. Remote-file windows can be declared through the COMS Utility, or dynamically created by executing a program that opens a remote file.

Access to a program environment through a window is called a dialog. COMS allows multiple and simultaneous dialogs with the same window during the same session. Each dialog is treated as an independent session with a program environment.

# Migrating Directly or Using a COMS MCS Window

When you migrate directly to COMS, you replace your current message control system (MCS) with COMS. If you decide to delay replacement of your MCS with COMS, you can establish a COMS MCS window in which to operate your MCS.

Your current MCS, the size and complexity of your current applications, available expertise, hardware, and the features and functions offered by COMS all influence your choice of a migration strategy.

# Migrating Directly to COMS

The direct approach consists of performing all required tasks to replace your current MCS or MCSs with COMS at one time. Depending on your MCS or MCSs, direct migration can mean simultaneously modifying your database, changing your application programs, and creating COMS entities.

For information on modifying your database and creating COMS entities, refer to the *COMS Configuration Guide*. The *COMS Programming Guide* contains information on how to write application programs for transaction processing that run under COMS.

Direct migration might involve replacing a user-created MCS or MCSs with COMS. For example, user-created dial-out MCSs can be replaced with COMS-supplied dial-out facilities. Remote printing MCSs can be replaced with Unisys Remote Print Systems (ReprintS). (Under ReprintS, the remote destination must be a data comm station controlled by MARC and COMS.)

Direct migration from GEMCOS to COMS involves the specific steps and considerations outlined in the paragraphs that follow. The format support library (FS-LIB) described in Section 3 can assist in your direct migration strategy.

## Migrating Directly from GEMCOS to COMS

If you currently use GEMCOS, you can migrate directly to COMS if you alter your application programs and COMS to emulate or accommodate the features of GEMCOS.

The following five steps outline the basic strategy of a direct migration from GEMCOS to COMS.

1. Install GEMCOS on a Mark Release, level 3.5 or later.
2. Install COMS and MARC.
3. Change the database restart data set to support COMS recovery.
4. Convert your GEMCOS application programs into programs that use COMS programming conventions.
5. Translate the network description in the GEMCOS transaction control language (TCL) control file into a network description that can be understood and used by COMS.

COBOL(68) programs do not need to be converted to COBOL74. However, COBOL74 has language support for the COMS interface. With COBOL(68) you must program the interface using LIBRARY calls.

## Translating a GEMCOS TCL Control File

For the direct migration to be successful, the network description specified in the TCL control file needs to be translated into a network description that can be understood and used by COMS. GEMCOS entities (categories of information in the control file) need to be translated into COMS entities.

Most GEMCOS entities have a direct COMS counterpart. For example, the GEMCOS entity called a SYSTEM is the counterpart of the COMS WINDOW entity.

The following table lists several GEMCOS entities and their COMS counterparts.

| GEMCOS | COMS |
|---|---|
| SYSTEM | WINDOW |
| PROGRAM | PROGRAM |
| INPUTQUEUE | AGENDA |
| STATION | STATION |
| DEVICE | DEVICE_TYPE |

The "GEMCOS Sources for COMS Entities and Attributes" appendix gives the GEMCOS counterpart for each COMS entity.

## GEMCOS Features in a COMS Environment

Before beginning a direct migration, read the "Comparing GEMCOS to COMS" appendix to see if the GEMCOS features you need are supported under COMS.

If you need another feature, consider creating your own processing item to support the feature.

## GEMCOS Formatting under COMS

The format support library available with COMS provides GEMCOS formatting and paging features in the COMS environment. The Screen Design Facility (SDF), another InterPro product, also can be used to provide formatting for devices compatible with ET 1100 terminals.

# Operating under an MCS Window

Under COMS, a window provides access to a program or system. MCS windows give you access to message control systems (MCSs). A site can have a maximum of 1023 windows. Only four windows (MARC, COMS Utility, CANDE, and GEMCOS) are defined at system initialization. You can define additional windows any time. You could define a window for each message control system supported at your site. Refer to Section 2 for more information on MCS windows.

## Accessing an MCS

When you access an MCS through a COMS window, the MCS and its application programs do not have to be modified. (This is a major difference from direct migration, in which the programs must be modified.) For example, CANDE and programs running under CANDE need no modifications to operate under a COMS window. However, as shown below, using an MCS window creates an extra processing layer.

## Performance under a COMS Window

There are no performance benefits when you run any MCS under a COMS window.

## Using the GEMCOS Window

The Kernel version of COMS enables a station to have one dialog with the GEMCOS window. The Kernel version does not offer the conveniences and benefits of the full-featured version of COMS. The full-featured version allows multiple and simultaneous dialogs with GEMCOS at one station and gives current GEMCOS users the windows and menu-driven features provided by COMS. However, the GEMCOS window should be used only as a short-term strategy in the migration to COMS.

Figure 1–1 shows what happens when you run GEMCOS under an MCS window. The extra layer, COMS itself, is used when any MCS operates under a window. This extra layer can hinder performance.

```
GEMCOS Alone:

   Application Program < --- > GEMCOS < --- > Hardware

GEMCOS under COMS:

   Application Program < --- > GEMCOS < --- > COMS < --- > Hardware
```

**Figure 1–1. Adding a Software Layer by Running GEMCOS under a COMS Window**

In most situations, you should use the GEMCOS window as an interim tool before you begin converting your GEMCOS application programs. However, CP 2000 users must use the GEMCOS window; GEMCOS cannot directly access CP 2000 stations. For more details about GEMCOS windows, read Section 2.

# Migration Assistance

If you have a migration solution that is not covered in this document or if you have created a migration tool that you would like to share with others, please send a detailed description to your local Unisys technical representative.

If you have a problem that requires immediate attention, contact your local Unisys technical representative.

# Section 2
# MCS Windows

This section describes the characteristics of a Communications Management System (COMS) message control system (MCS) window and indicates how it works. The section describes how COMS handles various DCWRITEs with respect to an MCS window, and presents DCWRITE-message and result-message formats. DCWRITEs are intrinsic functions that cause messages using these functions to be passed to the data communications controller (DCC) for action.

The information in this section pertains to a COMS MCS window in general, that is, a COMS MCS window that is used for any given message control system (MCS). For information about running the Generalized Message Control System (GEMCOS) in a COMS MCS window, refer to Section 4.

The following topics are presented in this section:

- General description of a COMS MCS window
- MCS-controlled pseudostations
- Naming conventions for pseudostations
- Requirements for an MCS to function through COMS
- Entering control commands
- Handling of DCWRITEs
- DCWRITE-message and result-message formats

For additional information about pseudostations and DCWRITEs, refer to the *DCALGOL Reference Manual.*

Unless indicated otherwise, the information in this section applies to a station or pseudostation controlled by an MCS operating either with the CP 2000 front-end processor through COMS or with the network support processor (NSP) through COMS.

Note that a given MCS operating with the NSP through COMS may have more capabilities than the same MCS operating with the CP 2000 through COMS.

Note also that an installation that includes both the CP 2000 and the NSP can have a mixed network in which stations and pseudostations controlled by an MCS can operate individually. These stations and pseudostations can function in any or all the following ways:

- With the CP 2000 through COMS
- With the NSP through COMS
- Directly under the NSP

*Note: The LSCANDE virtual-terminal protocol must be used for a terminal to operate with the CP 2000 through COMS.*

# General Description of a COMS MCS Window

A COMS MCS window is a feature that enables you to have a number of independent sessions (dialogs) with an MCS simultaneously at a single station, if that MCS is installed on the network. Examples of MCSs are the Command and Edit (CANDE) system, GEMCOS, and user-provided MCSs.

The Kernel version of COMS enables a station to have up to two dialogs of the CANDE window and one dialog of the GEMCOS window at the same time.

The full-featured version of COMS provides a means to define MCS windows through the COMS Utility program, as described in the *COMS Configuration Guide*. The COMS Utility can be used to define a number of MCS windows for one or more kinds of MCSs.

Once an MCS window is defined through the COMS Utility program for an installed MCS, you can have a number of simultaneous dialogs with the MCS at your station. The window can be defined through the COMS Utility to allow a maximum of eight simultaneous dialogs with the MCS at any station. You can then perform various activities associated with that MCS at the same time on one physical terminal.

In general, the windowing capabilities of COMS enable you to perform a variety of activities simultaneously on the terminal. These activities can be associated with one or any number of MCSs and other systems that have been installed and window-defined.

For information about controlling window dialogs at a terminal, refer to the *COMS Operations Guide*.

# MCS-Controlled Pseudostations

When you open a dialog of a COMS MCS window at your station, the dialog is established through an operating system feature called a *pseudostation*. A pseudostation is a virtual station that can be attached to and controlled by an MCS in much the same way as a real station. Unlike a real station, however, a pseudostation is not declared in the Network Definition Language II (NDLII), has no line assigned, and needs no corresponding physical terminal on the local host.

For every MCS window dialog that you open, COMS allocates a pseudostation and transfers it to the MCS in question. Each DCWRITE that the MCS performs on the pseudostation is handled in one of the following ways:

- Handled entirely by the operating system. The DCWRITE works as it would for a real station.
- Handled primarily by the operating system, with some interpretation by COMS.
- Interpreted by COMS.
- Rejected because it is not allowed on a pseudostation.

The pseudostation feature enables COMS to implement MCS windows. This feature does not apply to COMS direct windows or remote-file windows. When information is input through the MCS window dialogs at a given real station, the resulting output to the user is handled as if the response were to a number of real stations. In fact, only one terminal is involved.

In a COMS environment, the MCS or MCSs handling the input/output never come in contact with the name corresponding to the real station. Rather, the MCS or MCSs recognize the name of the pseudostation corresponding to each dialog.

You can recognize output associated with a particular MCS window dialog by means of the pseudostation name identifying that output.

# Naming Conventions for Pseudostations

Pseudostation names are assigned automatically by COMS. When your station is on the local host, the naming convention for pseudostations is as follows:

```
<NDLII station name>/<MCS window name>/<dialog number>
```

**Example**

```
ABC/CANDE/2
```

When your station has been transferred to another host through BNA, the naming convention for pseudostations is as follows:

```
<system host ID>/<NDLII station name>/<MCS window name>/<dialog number>
```

**Example**

```
A9A/ABC/CANDE/2
```

# Requirements for an MCS to Function through COMS

In general, an MCS that presently works with real stations on lines needs no modification to function in a COMS MCS window if the MCS satisfies all the following requirements:

- The MCS does not identify its stations using the DLS number format, shown below:

  ```
  <relative NSP number>:<line number>:<station number>
  ```

  Instead, the MCS identifies its stations using the logical station number (LSN).
- The MCS does not have to reconfigure the network and handle switched lines as part of the application.
- The MCS does not communicate with the data comm subsystem through toggles and tallies.

# Entering Control Commands

The following information is provided to help a station user communicating through a COMS MCS window to enter control commands directed to COMS or the given MCS.

If you enter a control command prefixed by one question mark (?), COMS always responds to the command, even if the command syntax is identical to that of a control command associated with the given MCS. If your entered control command prefixed by one question mark is not a COMS command, it is sent to the MCS.

If you enter a control command associated with the given MCS prefixed by two question marks (??), the command is always sent to the MCS, even if the command syntax is identical to that of a COMS control command.

# Handling of DCWRITEs

DCWRITEs are DCALGOL intrinsic functions that cause specified messages constructed to use these functions to be passed to the data communications controller (DCC) for action. The action that the DCC takes on a message depends on the type and variant fields of the message.

Information about the handling of DCWRITEs for a pseudostation in a COMS MCS window is presented in the paragraphs that follow. The information is presented in tables under the following categories:

- DCWRITEs handled by the operating system
- DCWRITEs interpreted by COMS
- Result messages given by COMS
- DCWRITEs not allowed by COMS

## DCWRITEs Handled by the Operating System

Table 2–1 lists the type numbers and corresponding functions for DCWRITEs that either are handled entirely by the operating system or primarily by the operating system for a pseudostation in a COMS MCS window.

COMS is notified of the DCWRITEs that are handled primarily by the operating system, and COMS takes some action on these DCWRITEs. In Table 2–1, these DCWRITEs are identified by the parenthetical note "(some COMS action taken)." The action that COMS takes on each of these DCWRITEs is indicated in Table 2–2.

**Table 2–1. DCWRITEs Handled by the Operating System**

| DCWRITE Type Number | DCWRITE Function |
|---|---|
| 00 | Initialize primary queue. |
| 01 | Attach station (some COMS action taken). |
| 02 | Interrogate MCS. |
| 03 | Communicate among MCSs. |
| 04 | Interrogate station environment. |
| 05 | Attach schedule station. |
| 32 | Change current queue (some COMS action taken). |
| 42 | Detach station. |
| 45 | Transfer station control. |
| 64 | Assign station to file (some COMS action taken). |
| 65 | Write to object job. |
| 66 | Cause station break. |
| 67 | Add station to file (some COMS action taken). |
| 69 | Subtract station from file. |
| 129 | Exchange clusters. |
| 131 | Update line attributes. |

## DCWRITEs Interpreted by COMS

Table 2–2 lists the type numbers and corresponding functions for DCWRITEs that are interpreted by COMS for a pseudostation in a COMS MCS window. The action that COMS takes on each of these DCWRITES is indicated in the table.

Note that the expression "giving dummy good result to MCS" in the table means that COMS is notifying the MCS that the function associated with the particular DCWRITE has been accomplished. The expression does not mean that COMS performs the function.

**Table 2–2. DCWRITEs Interpreted by COMS**

| DCWRITE Type Number | DCWRITE Function | COMS Action |
|---|---|---|
| 01 | Attach station. | Determine whether a good result is required by the MCS. |
| 32 | Change current queue. | Determine whether a good result is required by the MCS. |
| 33 | Write. | Send text; return the result to the MCS if wanted. |
| 34 | Read-once only. | No action. |
| 35 | Enable input. | Indicate that the input has been enabled by giving a dummy good result to the MCS. |
| 36 | Disable input. | Indicate that the input has been disabled by giving a dummy good result to the MCS. |
| 37 | Make station ready/not ready. | Resume or suspend window dialog. Make pseudostation ready or not ready. Give results to the MCS. |
| 38 | Set application number. | Indicate that the application number has been set by giving a dummy good result to the MCS. |
| 39 | Set characters. | Indicate that the characters have been set by giving a dummy good result to the MCS. |
| 40 | Set transmission number. | Indicate that the transmission number has been set by giving a dummy good result to the MCS. |

continued

Table 2–2. DCWRITEs Interpreted by COMS (cont.)

| DCWRITE Type Number | DCWRITE Function | COMS Action |
|---|---|---|
| 41 | Recall message for the following four situations: | |
| | COMS is recalling. | Give a result to the MCS indicating that no message has been recalled. |
| | MCS wants only first message. | Give a result to the MCS indicating that no message has been recalled. |
| | Object job output | Recall the first message. Give it to the MCS. |
| | Else | Recall all messages. Give them to the MCS. |
| 43 | Set/reset LOGICALACK (logical acknowledgement). | Indicate that LOGICALACK has been set or reset by giving a dummy good result to the MCS. |
| 44 | Acknowledge. | Indicate that acknowledgement has been made by giving a dummy good result to the MCS. |
| 46 | Write and return. | Send text; return the result if wanted. |
| 48 | Request null station. | Issue a null request (for NSP) or issue minor synchronization (for CP 2000). |
| 64 | Assign station to file. | Send text out if it applies to the message. |
| 67 | Add station to file. | Send text out if it applies to the message. |
| 68 | Change terminal attributes. | No action. |
| 96 | Make line ready. | Indicate that the line has been made ready by giving a dummy good result to the MCS. |
| 97 | Make line not ready. | Indicate that the line has been made not ready by giving a dummy good result to the MCS. |
| 103 | Set/reset line toggles/tallies. | No action. |
| 105 | Force line not ready. | No action. |

## Result Messages Given by COMS

Table 2–3 lists the values for and descriptions of DCWRITE result messages that COMS gives to the MCS for a pseudostation in a COMS MCS window. An explanation of how COMS handles the result for each of these result messages is indicated in the table.

**Table 2–3. DCWRITE Result Messages Given by COMS**

| Result Message Value | Message Description | Explanation |
|---|---|---|
| 00 | Good input | Message with text is input. |
| 01 | Station event | Only the control message variant of the input event type is given to the MCS. |
| 02 | File open | Not affected by COMS. |
| 03 | Object job output | Not affected by COMS. |
| 04 | File close | Not affected by COMS. |
| 05 | Good results | Good results are generated for all DCWRITEs that give 05 for a real station. |
| 06 | Recalled message | The message is recalled. |
| 14 | Station detached | The window is closed (that is, the pseudostation with the specified Logical Station Number (LSN) is deallocated). |
| 15 | Interrogate station | Pieces of information relating environment result to the pseudostation are returned. |
| 16 | Transfer station result | Not affected by COMS. |
| 18 | ODT-to-station result | Not affected by COMS. |
| 99 | Error result | An error result is generated only for the break case–that is, only for a break from a terminal device or printer. |

## DCWRITEs Not Allowed by COMS

Table 2–4 lists the type numbers and corresponding functions for DCWRITEs that are not allowed for a pseudostation in a COMS MCS window.

**Table 2–4. DCWRITEs Not Allowed by COMS**

| DCWRITE Type Number | DCWRITE Function |
|---|---|
| 49 | Set or reset sequence mode. |
| 53 | Write to transferred station. |
| 55 | Send result message to the MCS. |
| 56 | Set pseudostation attributes. |
| 98 | Dial out. |
| 99 | Disconnect. |
| 100 | Answer the phone. |
| 101 | Interrogate switched status. |
| 102 | Set or reset autoanswer. |
| 128 | Swap lines. |
| 130 | Move, add, or subtract station. |

If an attempt is made to execute a DCWRITE that is not allowed on a pseudostation in a COMS MCS window, one of the error messages listed in Table 2–5 is returned on the call.

**Table 2–5. Error Messages**

| Error Message Value | Description |
|---|---|
| 188 | That DCWRITE is not allowed on a pseudostation. |
| 190 | The pseudostation already has a fully participating MCS. Therefore, the requested type of transfer-station DCWRITE cannot be granted. |
| 192 | That DCWRITE is allowed only by a fully participating MCS. |

For a discussion of "full participation," refer to the *DCALGOL Reference Manual.*

# Formats of DCWRITE Messages and Result Messages

The formats of DCWRITE messages and result messages associated with particular DCWRITEs are presented in the paragraphs that follow. These formats are presented to show the way COMS handles fields in the header and in other words of these messages, such as the text.

## DCWRITE Message Format

The format of DCWRITE messages used in connection with DCWRITE 33 (WRITE) and DCWRITE 46 (WRITE and RETURN) is presented in Table 2–6. Fields in this message format are identified and described in the table. The table indicates which fields are handled by COMS, and specifies how some of these fields are handled. The table also indicates fields that are not handled by COMS. Brief explanations of the fields handled by COMS follow the table.

Additional information about the fields described in Table 2–6 is presented in the *DCALGOL Reference Manual.*

**Table 2–6. DCWRITE Message Format**

| Word/Field | Field Description | How Field is Affected by COMS |
|---|---|---|
| MSG[0].[47:08] | Type | Handled |
| MSG[0].[39:16] | Carriage control | Forwarded to the network |
| MSG[0].[23:24] | LSN | Mapped onto real station |
| MSG[1].[47:01] | (Not used) | |
| MSG[1].[46:07] | Priority | Not handled |
| MSG[1].[39:08] | Toggles | Not handled |
| MSG[1].[31:32] | (Not used) | |
| MSG[2].[47:08] | Retry count | Not handled |
| MSG[2].[39:16] | Text size | Handled |
| MSG[2].[23:24] | (Not used) | |
| MSG[3].[47:24] | (Not used) | |
| MSG[3].[23:24] | Tallies | Not handled |
| MSG[4].[47:24] | Message number | Saved for result message |
| MSG[4].[23:24] | (Not used) | |
| MSG[5] | (Not used) | |
| MSG[6]-MSG[N] | Text | Handled |

The type field (MSG[0].[47:08]) contains a value that tells the DCWRITE function which operation to perform.

The carriage control field (MSG[0].[39:16]) provides information about advancement of position in the output device.

The LSN field (MSG[0].[23:24]) designates the pseudostation to be affected by the DCWRITE function.

The text size field (MSG[2].[39:16]) specifies the number of bytes of text in the message minus the number of bytes in the message header.

The message number field (MSG[4].[47:24]) contains a value that identifies the message to the MCS.

The text field (MSG[6]-MSG[N]) contains the text of the message. This field is left justified and starts in word 6 of the message.

## Result Message Formats

Table 2–7 presents the format of result messages returned for DCWRITE 33 (WRITE) and DCWRITE 46 (WRITE and RETURN).

Table 2–8 presents the format of result messages returned for DCWRITE 41 (RECALL message).

In these tables, fields in each of the result message formats are identified and described, and the returned information is shown. Brief explanations of fields for which information is returned follow the tables.

Additional information about the fields described in Tables 2–7 and 2–8 is presented in the *DCALGOL Reference Manual.*

**Table 2–7. Format of Result Messages Returned for DCWRITEs 33 (WRITE) and 46 (WRITE and RETURN)**

| Word/Field | Field Description | Returned Information |
|---|---|---|
| MSG[0].[47:08] | Class | 05 |
| MSG[0].[39:16] | Variant | 00 |
| MSG[0].[23:24] | LSN | Pseudo-LSN |
| MSG[1] | | 00 |
| MSG[2] | | 00 |
| MSG[3] | | 00 |
| MSG[4].[47:24] | Message number | Value in original DCWRITE |
| MSG[4].[23:24] | Original DCWRITE | Original DCWRITE (33 or 46) |
| MSG[5] | | 00 |
| MSG[6]-MSG[N] | Text | Not returned |

**Table 2–8. Format of Result Messages Returned for DCWRITE 41 (RECALL Message)**

| Word/Field | Field Description | Returned Information |
|---|---|---|
| MSG[0].[47:08] | Class | 06 |
| MSG[0].[39:16] | Variant | 00 or 01 |
| MSG[0].[23:24] | LSN | Pseudo-LSN |
| MSG[1] | | 00 |
| MSG[2] | | 00 |
| MSG[3] | | 00 |
| MSG[4].[47:24] | Message number | Value in original DCWRITE |
| MSG[4].[23:24] | | 00 |
| MSG[5] | | 00 |
| MSG[6]-MSG[N] | Text | Not returned |

The class field (MSG[0].[47:08]) contains a value that determines the way in which the remainder of the message is interpreted.

The variant field (MSG[0].[39:16]) indicates qualification, variations, or additional information concerning the message type.

The LSN field (MSG[0].[23:24]) designates the pseudostation destination for the result message.

The message number field (MSG[4].[47:24]) contains a value that identifies the original DCWRITE to the MCS.

# Section 3
# Format Support Library

With Communications Management System (COMS) direct windows, you can process messages before they are received by programs on input (preprocessing) or before they are sent to a device on output (postprocessing). Processing is performed by processing items. The processing items must be defined in the COMS configuration file through the COMS Utility.

Processing items are grouped into processing-item lists. Processing-item lists are placed in agendas to determine which items to apply to messages that invoke the agendas on input or output. The agendas are also defined to route the messages to destination programs for the agenda.

The ability to direct preprocessing and postprocessing of messages makes a processing item an excellent tool to maintain Generalized Message Control System (GEMCOS) formatting and paging in the COMS environment. COMS provides a format support library (FS-LIB) with an ACTUALNAME of TCL_FORMATTER to maintain GEMCOS input formatting, output formatting, forms request, paging, and library formats.

The FS-LIB should be included in any COMS agenda that needs to perform GEMCOS message formatting. This processing item can be used in a direct migration approach and with the Screen Design Facility (SDF).

***Note: You cannot use the FS-LIB when running programs under any MCS window.***

This section covers the following topics:

- When and how to use the processing item
- Using the format support library with SDF
- GEMCOS formatting retained under COMS through the processing item
- Performance when using the processing item
- The logical flow of the processing item
- The needed COMS environment
- How to prepare the GEMCOS transaction control language (TCL) file to work in the COMS environment

## Why You Should Use the Format Support Library

The format support library (FS-LIB) is designed to be used in the direct migration from GEMCOS to COMS. It emulates GEMCOS formatting features in the COMS environment. The library enables you to maintain the formatting specified in the GEMCOS TCL file while running under COMS. If another feature or function is needed,

you can create your own processing item and include it in the appropriate agenda or agendas.

As described below, the FS-LIB can be used with the Screen Design Facility (SDF). You can use SDF instead of the processing item, or you can use SDF with the processing item.

# Using the Format Support Library with SDF

SDF is an InterPro product designed to create and maintain forms. SDF generates processing items that are used by COMS for formatting.

The format support library (FS-LIB) and the SDF processing items can be used at the same time. The SDF processing items should be placed after the FS-LIB in the agenda's processing-item list for both input and output messages.

If the FS-LIB does any formatting, it sets the first word of the program's conversation area to all 1s. SDF processing items examine the first word of the program's Conversation area on both input and output. If the word has all bits set to 1, SDF knows the FS-LIB has done the formatting; SDF does not process the data. If the word does not have all bits set to 1, processing continues and SDF formatting is performed.

# GEMCOS Formatting Retained in COMS

Formatting defines how a message is handled prior to its delivery to a program (input messages) or an output device (output messages). The format support library (FS-LIB) emulates GEMCOS formatting by performing the following functions:

- Formatting input and output messages
- Handling forms request
- Handling paging
- Supporting user-created format libraries

## Formatting Input and Output Messages

Stations are grouped by device class. By identifying the device class and either the message key or message ID, the FS-LIB applies the proper format to a message.

## Making a Forms Request

A forms request is a request for a blank form. When the request is processed, the terminal displays only prompts and format-supplied defaults; no previously entered values are shown. To make a forms request, a station transmits an output message ID with no data and no special terminating character.

## Dividing Messages into Pages

Paging is dividing a message into distinct segments. This is usually done when

- The message is too long or crowded for the entire message to be displayed on the terminal at one time.
- The programmer wants to make specific distinctions between parts of the message.

A message can be divided into two or more pages. A given message can be paged at one device type and not paged at another device type. Both input and output messages can be paged.

Because the FS-LIB operates as a processing item, paging is applied separately on each SEND executed by a program. Therefore, paging cannot be applied across all segments of a segmented message.

When a station enters paging mode, COMS examines the first character of transmitted data to determine what action to take. The same characters that control the GEMCOS paging dialog also control the COMS FS-LIB paging dialog.

## User-Created Formats

The GEMCOS Format Library interface enables users to create message formats by using ALGOL, COBOL(68), or COBOL74. These formats are maintained in addition to the standard TCL formats. All the user-created formats must be maintained in one format library. The title of the format library must be specified in the global section of the TCL file. If no title is specified, the default title is GEMCOS/FORMATLIBRARY.

# Using the Format Support Library

All GEMCOS formats for input, output, forms requests, paging, and library are retained in COMS through the use of the format support library (FS-LIB).

The following paragraphs discuss various aspects of the FS-LIB.

## Using the Format Support Library in Agendas

The FS-LIB must be included in an agenda to perform input or output formatting, a forms request, or paging. If SDF processing items are also used, the SDF processing items must be included after the FS-LIB.

When an agenda is used for output formatting, the agenda should not be designated as the default output agenda for that window. Otherwise, a stack overflow error could occur when the FS-LIB attempts to report errors to the monitor stations.

## Formatting Performance Expectations

Using the FS-LIB instead of GEMCOS could improve your system's formatting performance because the library runs as part of the COMS multiprogramming environment, whereas GEMCOS uses single-streamed formatting. At minimum, your system should format as efficiently using the FS-LIB as it does using GEMCOS.

## Logical Flow for Input and Output Formatting

When input or output formatting is required, the format support library follows the basic logical flow shown below:

1. Determines the window where the message was entered or is being sent and locates the appropriate TCL system.
2. Gets the device type for the station where the input message was entered or where the output message is to be sent.
3. Searches for the proper format using the system index, the device type, and either the format ID or message key.
4. Establishes parameters and calls the formatter routine to format the message.
5. Exits.

## Logical Flow for Forms Request

When a forms request is entered, the FS-LIB varies the basic logical flow. Steps 1, 2, and 5 of a forms request are the same steps used for input and output formatting. Steps 3 and 4 are different.

1. Determines the window where the message was entered or is being sent, and locates the appropriate TCL system.
2. Gets the device type for the station where the input message was entered or where the output message is to be sent.
3. For input messages with lengths less than or equal to six characters, searches among all message IDs to see if the user is requesting a form.
4. For a forms request, sends the form to the requesting station.
5. Exits.

## Logical Flow for Paging

When a user application program sends a message that requires paging, the basic logical flow followed by input and output formatting is initiated. At step 4, the FS-LIB calls paging routines to process the message. Steps 4, 5, and 6 are directed by the paging routines.

1. Determines the window where the message was entered or is being sent, and locates the appropriate TCL system.
2. Gets the device type of the station where the input message was entered or where the output message is to be sent.
3. Searches for the proper format using the system index, the device type, and either the format ID or message key.
4. For paged formats, calls paging routines to send the first screen to the device and to prepare for paging dialog.
5. For an update paged format, when the user sends the paging complete character (X), sends the entire formatted message to COMS as a new input.
6. Exits.

Formatting is then handled by paging routines until the paging mode is exited.

With the FS-LIB, there is no restriction on the number of stations that can use paging at one time even if GEMCOS control paging is specified in the TCL.

# Creating the COMS Environment

To produce the correct formatting, the format support library (FS-LIB) requires the following COMS entities:

- Windows
- Trancodes
- Device types
- Libraries
- Processing items
- Processing-item lists
- Agendas

Create these entities through the COMS Utility before using the FS-LIB. (See the "GEMCOS Sources for COMS Entities and Attributes" appendix for a listing of GEMCOS and COMS counterparts.)

Because of the importance of the COMS entities listed above to the FS-LIB, information about defining them to COMS is presented in the paragraphs that follow. The COMS Utility command for creating each of these entities is included. Also presented is

information about updating formats, amending application programs to use the COMS direct interface, recognizing input formatting errors, and obtaining monitor output.

## Creating COMS Windows for Systems in the GEMCOS TCL

It is possible for more than one COMS direct-window to access all the formats of the FS-LIB. This capability is accomplished by setting the GEMCOS global attribute MASTERCOMPUTE to TRUE and having only one system declared in the GEMCOS TCL. Then the FS-LIB accepts any format request from more than one COMS direct window.

However, you might want to restrict format requests to specific windows. If MASTERCOMPUTE is equal to FALSE, or if there are multiple systems in the TCL, access to the FS-LIB is established as follows.

For every system in the GEMCOS TCL for which you want to use formatting in COMS, create a window with the same name as the system. In the TCL, at least one program must be declared in a system. All message keys for a system must be declared under that program.

The COMS Utility command for creating a window is the following:

```
CREATE WINDOW <window name>
```

A window name can be 1 to 17 alphanumeric characters.

## Defining Trancodes

The COMS Utility command for defining a trancode to COMS is shown below. Ensure that the trancode (message key) from the TCL matches the input in the command.

```
CREATE TRANCODE <trancode name> OF <window name>
AGENDA = <agenda name>
FUNCTION = <integer>;
```

A trancode name and a window name can be 1 to 17 alphanumeric characters. An agenda name can be 1 to 17 alphanumeric characters and must begin with a letter. The integer for FUNCTION can be a number from 0 (zero) to 9999.

Note that all trancodes must start in position one of the input.

***Note:*** *The FS-LIB discontinues (DSes) the task if a conversation area is not allocated in the header of the program.*

## Defining Device Types

The COMS Utility command for defining a device type to COMS is shown below. Ensure that the device declared in the TCL matches the input in the command.

```
CREATE DEVICE_TYPE = <device_type name>;
CREATE STATION <station name>
       DEVICE_TYPE = <device_type name>;
```

A device type name can be 1 to 17 alphanumeric characters and must begin with a letter. A station name can be composed of up to 12 nodes. Each node must start with a letter or a number, followed by up to 16 additional characters. These succeeding characters can be letters, numbers, underscores (_), or hyphens(-). The station name can be a maximum length of 255 characters.

## Defining Processing-Item Libraries

In COMS, a library contains a set of procedures called processing items. These items can be called by agendas to process messages before or after input or output. A library name must be assigned to each processing item defined in the configuration file.

The COMS Utility command for defining a processing-item library to COMS is shown in the following:

```
CREATE LIBRARY <library name>;
```

A library name can be 1 to 17 alphanumeric letters and must begin with a letter.

You can have multiple copies of the FS-LIB to process different TCL files. Each copy must be a separate copy of the code file, and each can have one GEMCOS FORMATLIBRARY.

## Defining Processing Items

The processing item ACTUALNAME must be TCL_FORMATTER. It must be associated with a previously defined processing-item library.

The COMS Utility command for defining a processing item to COMS is shown in the following:

```
CREATE PROCESSING_ITEM <processing-item name>
       ACTUALNAME = TCL_FORMATTER,
       LIBRARY = <library name>;
```

A processing-item name and a library name can be 1 to 17 alphanumeric letters and must begin with a letter.

## Creating Processing-Item Lists

The processing-item list contains the processing items to be called by the agenda before it processes the message. The order in which the processing items appear in the list is the order in which they are called. If the FS-LIB is used in conjunction with SDF, the SDF processing items must follow the FS-LIB processing item.

Include the FS-LIB in an agenda's processing-item list when any input or output formatting, forms request, or paging is required. Include it in the default agenda for windows that might use paging. When COMS encounters invalid trancodes, such as paging dialog characters, the default agenda is invoked.

The COMS Utility command for creating a processing-item list is shown in the following:

```
CREATE PROCESSING_ITEM_LIST
       <processing_item_list name> =
          <processing_item name1, name2, name3,...>;
```

A processing-item list name can be 1 to 17 alphanumeric characters and must begin with a letter.

## Using Agendas When Formatting

A COMS agenda processes and routes messages on input, output, or both, for direct windows. To use an agenda on input, the agenda must have an assigned destination program. Form requests and paging dialog messages, however, do not go to the destination program. The destination is specified either in the output header of the message or in the output header of the application program. By default, the destination is assumed to be the input station.

To use update paging with a particular window, the window must have an additional agenda created with INPUT_ROUTER as its destination. This agenda must have the same name as the window and the GEMCOS TCL system.

The COMS Utility command for defining an agenda to COMS is shown in the following:

```
CREATE AGENDA <agenda name> OF <window name>
       PROCESSING_ITEM_LIST = <processing_item_list name>,
       DESTINATION = <program name>;
```

An agenda name, processing-item list name, and a program name can be 1 to 17 alphanumeric characters and must begin with a letter. A window name can be 1 to 17 alphanumeric characters.

## Updating Formats

To update formats, run the GEMCOS Utility against the TCL. Then disable the library by entering the following command:

```
?DISABLE LIBRARY <library name>
```

For more information about updating formats, refer to the *COMS Operations Guide.*

## Amending Application Programs

Application programs must be amended to use the COMS direct interface. This means that for output formats, the FORMID must be placed in the first six bytes of the COMS conversation area.

## Recognizing Input Formatting Errors

Input formatting (station to program) errors are reported to the application program by the conversation area. If a format error occurs, the second word of the COMS conversation area is set to 1. If a program's conversation area is less than two words long, input formatting errors are not reported to that program.

## Obtaining Monitor Output

The FS-LIB sends format exception notification to the COMS monitor stations. This notification is directed to the window on the monitor station that corresponds to the window on which the exception occurred at the user station. Thus, if notification is desired, the monitor station must have the same window name (the GEMCOS system name) declared, and that window must be open.

# What the GEMCOS TCL File Must Contain

The GEMCOS TCL file must be compiled, without any syntax errors, with the GEMCOS/UTILITY, release 8.0 or higher. (A Series GEMCOS users need not recompile their TCL. The format support library (FS-LIB) can be used with existing GEMCOS files as long as the files were generated by the GEMCOS/UTILITY, release 8.0 or higher.) If the release levels are incorrect at run time, an error is displayed and the FS-LIB goes to end of task (EOT) without being used.

The FS-LIB needs information from the device and format sections in the GEMCOS TCL file. In addition, it needs to know the system names defined in the TCL file so that the correct format can be applied according to which window the user is on. (This information is obtained from the files produced by the GEMCOS/UTILITY.)

The following files must be file-equated when the GEMCOS/UTILITY is run. The variable <codefile name> is the code file name of the FS-LIB. These files must have a security classification of PUBLIC I/O.

| GEMCOS File | File-Equated to |
|---|---|
| IQUS | <codefile name>/INPUT |
| OQUS | <codefile name>/OUTPUT |
| CQUS | <codefile name>/CONTROL |
| FMTFILE | <codefile name>/FORMAT |

The COMS FS-LIB provides that all the formats in a GEMCOS TCL file are made RESIDENT if PAGINGPERMANENT is assigned the value TRUE in the global section of the TCL file. If you have been running GEMCOS with PAGINGPERMANENT as TRUE to keep the GEMCOS PAGER stack in the mix, you should consider that PAGINGPERMANENT has a different function when running through COMS. You might want to assign the value FALSE to PAGINGPERMANENT if you do not want all formats to be RESIDENT.

# Section 4
# GEMCOS Windows

This section presents guidelines for running the Generalized Message Control System (GEMCOS) in a Communications Management System (COMS) message control system (MCS) window, and considers various aspects of GEMCOS operation through COMS. The section indicates limitations to GEMCOS features when GEMCOS is operating with the CP 2000 front-end processor through COMS.

The information in this section pertains specifically to running GEMCOS in a COMS MCS window. For information about a COMS MCS window in general, refer to Section 2.

The following topics are presented in this section:

- General description of a GEMCOS window
- Advantages of running GEMCOS through COMS
- Changing station names in GEMCOS transaction control language (TCL)
- Entering control and system commands
- GEMCOS feature limitations with the CP 2000
- Relative GEMCOS functional behavior with the CP 2000 and the network support processor (NSP)

Unless indicated otherwise, the information in this section applies to GEMCOS operating either with the CP 2000 through COMS or with the network support processor (NSP) through COMS. Note that the information in this section under "GEMCOS Feature Limitations with the CP 2000" applies only to GEMCOS operating with the CP 2000 through COMS.

## General Description of a GEMCOS Window

An MCS window that is used to run GEMCOS through COMS is called a GEMCOS window.

The Kernel version of COMS enables a station to have one dialog of the GEMCOS window.

The full-featured version of COMS provides a means to define a GEMCOS window that allows multiple simultaneous dialogs with GEMCOS at a station, once the GEMCOS system has been installed on the network. Such window definition is accomplished through the COMS Utility program, as described in the *COMS Configuration Guide*. A GEMCOS window can be defined to allow a maximum of eight simultaneous dialogs with the GEMCOS system at a station.

A number of GEMCOS windows can be defined through the COMS Utility. If the network is to include more than one GEMCOS system, a GEMCOS window can be defined and established for each of the systems.

# Advantages of Running GEMCOS through COMS

A distinct advantage of running GEMCOS through COMS, rather than running GEMCOS directly under the NSP—or directly under the data communications data link processor (DCDLP)—is your ability to have multiple simultaneous dialogs with GEMCOS. This ability allows efficient use of a terminal.

Another advantage of running GEMCOS through COMS is the convenient menu-driven transaction-processing facility provided to COMS users by the Menu-Assisted Resource Control (MARC) program. For information about the MARC program, refer to the *MARC Operations Guide.*

# Changing Station Names in GEMCOS TCL

When GEMCOS is installed on the same network with COMS, and a GEMCOS window is defined through the COMS Utility program, the names of the stations in the TCL for the GEMCOS-controlled environment must be changed. The reason for changing the names of these stations is to make the names compatible with the COMS naming convention for pseudostations. In other words, the name of each station in the TCL must be made the same as the name of the corresponding pseudostation allocated in COMS so that GEMCOS can identify which pseudostations it can communicate with and control.

## For One Dialog

To be compatible with the COMS naming convention for pseudostations, you must change the Network Definition Language II (NDLII) name of each station in the TCL to reflect the following information and format:

```
<NDLII station name>/<MCS window name>/<dialog number>
```

This change typically involves adding a window name and dialog number to each of the station names that already exist in the TCL. For example, assume that the TCL includes a station named STA1. If an MCS window named GEMCOS were declared for station STA1 through the COMS Utility, it would be necessary to change the name of the station to STA1/GEMCOS/1 in the TCL.

## For More Than One Dialog

To use more than one dialog, you duplicate the description of STA1/GEMCOS/1 in the TCL as many times as required, and make the corresponding changes to the dialog number.

For example, to use three dialogs from the station, you duplicate the description of station STA1/GEMCOS/1 in the TCL two times, and change the dialog number to 2 and 3. The result is three stations declared in the TCL (instead of the former single station STA1). You must also make all changes to device type and other areas in the TCL for the two new stations, to correspond to the way the original station was declared in the TCL. Consider that GEMCOS views each dialog as a different station, and makes no association between dialogs.

When GEMCOS is installed, a GEMCOS window is defined, and the station names are changed in the TCL, you then can have a number of simultaneous dialogs with GEMCOS at your station.

# Entering Control and System Commands

The following information is provided to help a station user communicating through a GEMCOS window enter control commands directed to COMS or GEMCOS. Information is also provided to help a system user who is communicating through a GEMCOS window enter system commands.

When GEMCOS is operating with the CP 2000 through COMS, data comm network control for GEMCOS is handled by COMS instead of by GEMCOS. Therefore, refrain from using GEMCOS commands to manage the network; instead, use the corresponding COMS commands.

When GEMCOS is operating with the NSP through COMS, data comm network control for GEMCOS is still handled by GEMCOS. Therefore, you can use GEMCOS commands to control parts of the network. Refer to "Requirements for an MCS to Function through COMS" in Section 2 for information.

You can enter GEMCOS commands not associated with network control at any time while you are communicating through a GEMCOS window, whether COMS is operating with the CP 2000 or the NSP.

To direct a command to COMS from a GEMCOS window, enter the command prefixed by one question mark (?). To direct a command to GEMCOS from a GEMCOS window, enter the command prefixed by two question marks (??). To enter a system command from a GEMCOS window, enter the command prefixed by three question marks (???).

Be sure to prefix your command with the correct number of question marks.

If you enter a command prefixed by one question mark (?), COMS attempts to handle the command, even if it is a GEMCOS command. If COMS understands the command, it handles it. If not, it passes the command to GEMCOS.

GEMCOS handles all commands prefixed by two question marks (??). Therefore, to ensure that GEMCOS handles a command, enter it prefixed by two question marks (??). For example, both COMS and GEMCOS have a STATUS command. If STATUS is prefixed by one question mark (?), COMS responds with the status. If STATUS is prefixed by two question marks (??), GEMCOS responds with the status.

Note that a command like CHANGE is always handled by GEMCOS, whether prefixed by one or two question marks. Since there is no CHANGE command in COMS, COMS gives the CHANGE command prefixed by one question mark (?) to GEMCOS.

# GEMCOS Feature Limitations with the CP 2000

Operating GEMCOS with the CP 2000 through COMS entails a number of limitations to GEMCOS features. These limitations are described in the following paragraphs.

## Limitations Due to Loss of Toggles and Tallies

Toggles and tallies are the station-oriented variables in the NDLII program. When GEMCOS is operating either directly under the NSP or with the NSP through COMS, GEMCOS uses toggles and tallies to communicate with the NSP.

The CP 2000 has built-in request sets and does not recognize toggles and tallies. Therefore, GEMCOS cannot use toggles and tallies when it is operating with the CP 2000 through COMS.

The inability of GEMCOS to use toggles and tallies when it is operating with the CP 2000 through COMS results in the following limitations to GEMCOS features:

- The STAMASTER attribute in the station section of the GEMCOS TCL is not supported. Since GEMCOS can no longer use the STAMASTER attribute to designate a station to be a master station, no computer connection that uses the STAMASTER attribute is supported.
- If an object job is used, all stations to which the object job is attached must have MYUSE equal to 3 (that is, all stations must be input- and output-capable). GEMCOS is unable to inform the NSP that it is sending a message to a network-control station on object job output.
- The BROADCAST attribute in the station section of the GEMCOS TCL is not supported. Therefore, GEMCOS cannot use the BROADCAST attribute to instruct the NSP to broadcast a message to all stations on a line. Note that a message can still be broadcast to all stations on a line, but the message is sent to each station one at a time rather than simultaneously to all stations.
- When the MSG-QBYPASS bit (COMMON[7].[46:1]) is assigned the value 1 by a user program, the output message is not audited. Unisys recommends that the MSG-QBYPASS bit (COMMON[7].[46:1]) be equal to 0 (zero). If the bit is equal to 1 and a station gives negative acknowledgement (NAK) to an output message, the message is lost.
- The NOQUEUE attribute in the station section of the GEMCOS TCL is not supported.

## Other Limitations

When GEMCOS is operating with the CP 2000 through COMS, limitations to GEMCOS features other than the limitations due to the loss of toggles and tallies are the following:

- The NOLINE and IDENTIFY attributes in the station section of the GEMCOS TCL are not supported because COMS, and not GEMCOS, handles the dial-in stations. Therefore, you cannot use a dial-in station to dial in to GEMCOS.
- The LOGICALACK attribute in the station section of the TCL in GEMCOS is not supported. Therefore, GEMCOS cannot use the NSP LOGICALACK (logical acknowledgement) function.
- The SETMCSREQUESTSET attribute in the station section of the GEMCOS TCL is not supported. Therefore, GEMCOS cannot use the SETMCSREQUESTSET attribute to instruct the NSP to switch between request sets.
- Process programs associated with GEMCOS are no longer able to issue network control commands to GEMCOS to control the data comm network. In this respect, the process programs are limited in the same way that station operators are limited in the ability to control the network. All network control is handled by COMS.

# Relative GEMCOS Functional Behavior with CP 2000 and NSP

Because of the limitations described under "GEMCOS Feature Limitations with the CP 2000" in this section, GEMCOS operating with the NSP through COMS is expected to perform more functions and provide more results than GEMCOS operating with the CP 2000 through COMS. However, GEMCOS operating with the CP 2000 through COMS is still expected to have most of the capabilities of GEMCOS operating with the NSP through COMS.

Further, GEMCOS operating with the NSP through COMS is expected to have almost all the capabilities of GEMCOS operating directly under the NSP.

# Section 5
# Remote Files

This section compares the handling of remote files through the Communications Management System (COMS) with the handling of remote files through the Command and Edit (CANDE) system. CANDE uses logical I/O to handle remote files, while COMS also has extensions for the window handling. Emphasis is placed in this section on how COMS handles remote files.

For a detailed discussion of remote files in a COMS environment, refer to the *COMS Programming Guide* and the *COMS Configuration Guide.*

## What is a Remote File?

A remote file is a file with a KIND attribute specified as REMOTE. A remote file enables object programs to communicate interactively with a data comm station.

You typically use a remote file when you want to dedicate a program to a certain user. A remote-file program characteristically involves data entry or inquiry on records from a station.

## Situations for Using Remote-File Windows

In operating under COMS, there are basically two situations in which you would consider using remote-file windows:

- If you have existing programs that use an existing remote-file interface
- If you want to dedicate a program to a certain user

## Comparison of COMS and CANDE Remote-File Handling

If you are operating under CANDE and execute a remote-file program, the remote-file program declares a remote file that by default works only with the single station that it recognizes. In other words, a remote-file program executed under CANDE can have only one user, unless it explicitly takes action to add more stations to its remote file. Separate copies of the program are needed for other users.

Under COMS, on the other hand, you can have remote files that accommodate a number of stations (multistation remote files) as well as single-station remote files.

When you run a remote-file program under CANDE, either in a CANDE window under COMS or through CANDE directly controlling your real station, the remote file takes

over the station, and all input except control commands is routed to the remote file. This is also true if you run a remote-file program outside of CANDE—for example, through Work Flow Language (WFL)—provided the remote file is label-equated to your station.

When you run a remote-file program under COMS, either by running it in the Menu-Assisted Resource Control (MARC) interface or by running it outside of COMS/MARC with the remote file label-equated to your station, COMS creates for it a dynamic remote-file window called "REM<number>." When the program is run from MARC, you can "move" to the remote file either by specifying *TASK* as a command to MARC or by specifying the following:

```
?ON REM<number>/<dialog number>
```

When the program is run outside of COMS/MARC, you have only the option of entering "?ON REM<number>/<dialog number>." When the dynamic remote-file window is the current window, all input except control commands is routed to the remote file.

A program that runs in CANDE can typically run under COMS as a dynamic remote-file program without any change.

Note that operating a remote-file program from a station under CANDE in a COMS MCS window is no different from operating a remote-file program from a station controlled directly by CANDE.

## Dynamic and Declared Remote Files

CANDE handles only dynamic remote files. COMS handles both dynamic remote files (that is, not defined to COMS) and declared remote files (that is, defined through the COMS Utility). Under COMS, dynamic remote-file programs are not shared. However, declared remote-file programs can be shared as defined through the COMS Utility.

A declared remote-file window is a window defined for an associated remote-file program through the COMS Utility. A declared remote-file program can handle either one station or a number of stations. The number is specified through the COMS Utility. Specific logic is normally required in the program to handle more than one station. A program that runs in CANDE can run under COMS as a declared remote-file program, but this declared program typically is not written to handle more than one station.

Note that if a remote-file program receives parameters or requires a run-time file equation or task equation, it cannot be used as a declared remote-file program under COMS. Moreover, remote-file programs that access attributes of the remote file might require modifications to work properly as declared remote-file programs.

## Multistation Remote Files

Under CANDE, a remote-file program can access more than one station in a remote file by either of two means. It can add stations to the file and subtract stations from the file by using the STATIONLIST file attribute as described in the *I/O Subsystem Programming Guide,* or it can open a file using a title of a file described in the installation Network Definition Language II (NDLII).

When COMS is used, a multistation remote-file program does not have to use either of the means indicated above to add and subtract stations. COMS does all the necessary adding and subtracting. When you first access the window (using *?ON <declared remote-file window>*), COMS adds the station to the file without help from the program. When you close the window (using *?CLOSE* or by logging off) COMS subtracts the station from the file. In this case, the remote-file program is informed of the closure by getting an end-of-file (EOF) indication for the relative station number (RSN) of the station.

## Input Remote Files

Under CANDE, you can have only one input remote file. Under COMS, you can have more than one input remote file, each in a separate dynamic remote-file window.

## Defining and Executing Declared Remote-File Programs

Because CANDE handles only dynamic remote files, you cannot define remote-file programs through CANDE.

Through the COMS Utility, you can define certain remote-file programs to COMS, including user-written programs that meet simple restrictions. You can then assign these programs unique window names, and make the windows available to all or selected users. These are declared remote-file windows. A user can then specify *?ON <name of declared remote-file window>* and execute the program. For information on how declared remote-file windows are defined through the COMS Utility, refer to the *COMS Configuration Guide.*

## Executing a Remote-File Program by a RUN Command

In a COMS environment, any dynamic remote-file program that you execute by a RUN command from a CANDE window or a MARC window opens a remote file to your station that shares the original window. No remote-file window is involved for programs executed in this manner.

## Closing a Remote File

Under COMS, remote-file programs receive an end-of-file (EOF) indication on their remote files whenever the window associated with the program is closed. If the LASTSUBFILE file attribute is interrogated at this point, it specifies the relative station number (RSN) of the station removed from the remote file to indicate that CLOSE has occurred for that station. Declared remote-file programs receive an additional EOF indication when COMS instructs the program to go to end of job (EOJ). In this case, the LASTSUBFILE attribute is assigned the value "RSN 1". This occurs when all users have closed a remote-file window and the minimum number of copies for that program has been exceeded.

The following program fragment is an example of the statements that might be used in a multistation remote-file program written in ALGOL and running under COMS. The fragment shows how the program handles an EOF on its remote file when the window associated with the program is closed.

```
DEFINE
  ERROR_FLAG = [Ø:1] #,
  EOF_FLAG   = [9:1] #;

WHILE TRUE DO
  BEGIN
  READBOOL := READ(RMT, 288, IN_BUF);
  IF NOT READBOOL.ERROR_FLAG THEN
    PROCESS_INPUT(IN_BUF)
  ELSE
    IF READBOOL.EOF_FLAG THEN
      IF RMT.LASTSUBFILE = 1 THEN
        WRAP_UP_PROGRAM
      ELSE
        HANDLE_CLOSE_OF-STATION
      ELSE
        .
        .
        .
    END;
```

The following program fragment is an example of the statements that might be used in a remote-file program written in COBOL and running under COMS. The fragment shows how the program handles an EOF on its remote file when the window associated with the program is closed.

```
77 LAST-STA-SW          PIC 9     VALUE Ø.
   88 LAST-STA                    VALUE 1.
   .
   .
MAINLINE.
   .
   .
   .
   PERFORM RECEIVE-INPUT THRU RECEIVE-INPUT-EXIT
           UNTIL LAST-STA.
   PERFORM WRAP-UP-PROGRAM.
   STOP RUN.

RECEIVE-INPUT.
  READ RMT;
     AT END
        PERFORM HANDLE-EOF
        GO RECEIVE-INPUT-EXIT.
  PERFORM PROCESS-INPUT.
RECEIVE-INPUT-EXIT.
  EXIT.

HANDLE-EOF.
  IF ATTRIBUTE LASTSUBFILE OF RMT = 1 THEN
     MOVE 1 TO LAST-STA-SW
  ELSE
     PERFORM HANDLE-CLOSE-OF-STATION.
```

## Tanking

Logical I/O, which is used by CANDE, does all the tanking for a remote file on a file basis. COMS, on the other hand, does all the tanking on a dialog basis for each station in a remote file. This means that COMS cannot apply the same rules for suspending output as logical I/O does. Logical I/O allows as many outstanding messages as there are file buffers for the file before suspending output. COMS allows 2400 characters per station, including data comm headers, to be outstanding.

## Handling of Time Limit on a Write Operation

Time-limit handling by logical I/O differs from time-limit handling by COMS. Logical I/O, which is used by CANDE, gives a time-limit exception when no file buffers are available for the write operation for the specified time period. Except as noted below, COMS gives a time-limit exception when the number of outstanding characters for a station has exceeded the limit of 2400 for the specified time period.

Note that if a broadcast write operation is done, COMS gives a time-limit exception only if no station in the remote file has been sent output.

## Direct I/O

Direct I/O on remote files is supported by logical I/O only if the controlling MCS, typically CANDE, controls the network support processor (NSP) stations directly. Direct I/O on remote files is not supported by COMS, nor is it supported in a COMS MCS window.

If a program that uses direct I/O on remote files is run in a COMS remote-file window, COMS handles the remote file as if it were a nondirect file with tanking set to ASYNC. In handling this I/O on the pseudostation for the MCS window, the operating system completes the I/O when COMS gets the transaction, not when the transaction is sent to the terminal.

# Appendix A
# Comparing GEMCOS to COMS

Appendix A consists of seven tables that compare the functions and features supplied by the Generalized Message Control System (GEMCOS) with those provided in the full-featured Communications Management System (COMS).

Each table compares one functional area. The tables are presented in alphabetical order:

- Table A–1. Application Program Interface
- Table A–2. Computer-to-Computer Communication
- Table A–3. Environment Interface
- Table A–4. Message Formatting
- Table A–5. Recovery
- Table A–6. Security
- Table A–7. Station Interface

Within each table, the features are listed alphabetically. Aspects of the feature are shown in alphabetical order.

Several GEMCOS features, while not supplied in COMS, can be implemented through a processing item or station list. When applicable, methods for implementing these features are noted in the tables.

Table A–1 compares the application program interface of GEMCOS to COMS.

**Table A–1. Application Program Interface**

| GEMCOS Feature | COMS Feature |
|---|---|
| Batch job interface | No. (Synchronized recovery with batch jobs is available outside COMS.) |
| Conversation areas | Yes. (Available on input to the program from a processing item.) |
| Editor program | No. (Can be implemented by a user-created processing item.) |
| Module/function indexes | Yes |
| Multiple input queues per program | No |
| Multiple transactions per execution | Yes |
| Object job interface: | |
| Program | No |
| Station | Yes |
| Practice mode | No |
| Priority scheme for input queues | No |
| Program initiated by NOINPUT command | No |
| Program can specify TITLE on output to indicate message sent to multiple stations. | No. (Can be implemented by a user-created processing item or a station list.) |
| Retry counter for Transaction Processor (TP) failure | Yes |
| Sending multiple output messages per transaction | Yes |
| Sending/receiving parts of messages | No. (Can be implemented by a user-created processing item or a station list.) |
| Service program | No. (Can be implemented by a user-created processing item or a station list.) |
| Single transaction per execution | No |
| Station bits | Yes. (COMS Installation data feature provides the same functionality.) |
| Test program | No |

continued

**Table A–1. Application Program Interface** (cont.)

| GEMCOS Feature | COMS Feature |
|---|---|
| Transaction-based routing as follows: | |
| Program-to-program response to program | No. (Can be implemented by a user-created processing item.) |
| Program-to-program response to station | Yes |
| Station to program | Yes |
| User-supplied routing | No. (Can be implemented by a user-created processing item.) |
| Transmission of messages to the following: | |
| Area broadcast | No. (Can be implemented by a user-created processing item or a station list.) |
| Area rotary | No |
| Monitor station | No. (Can be implemented by a user-created processing item or a station list.) |
| Network control station | No. (Can be implemented by a user-created processing item or a station list. |
| Originator | Yes |
| Program by number | Yes. (Sent by program agenda.) |
| Program by trancode | Yes |
| Station by name | No. (Can be implemented by a user-created processing item or a station list.) |
| Station by number | No. (Can be implemented by a user-created processing item or a station list.) |
| Stations by output message ID | No. (Can be implemented by a user-created processing item or a station list.) |
| Validation of area name | No |
| Validation of station name | Yes |

Table A–2 compares GEMCOS features to COMS features in a computer-to-computer communication.

**Table A–2. Computer-to-Computer Communication**

| GEMCOS Feature | COMS Feature |
|---|---|
| GEMCOS-to-GEMCOS file transfer | No |
| Routing headers for the following: | |
| Station GEMCOS-to-GEMCOS communication of transactions and output messages | No |
| Station as port file | No |
| Station that is port file | No |
| Transaction-based routing station to station by trancode | No |

Table A–3 compares GEMCOS features to COMS features in an environment interface.

**Table A–3. Environment Interface**

| GEMCOS Feature | COMS Feature |
|---|---|
| Administrative message switching | No |
| COMPUTE command | No |
| Control of the following: | |
| Database programs | Yes |
| Specified stations | Yes |
| Dynamic volume | Yes |
| Interception of station message traffic | No. (Can be implemented by a user-created processing item.) |
| Limitation of memory for program input messages | Yes |
| Memory subsystem control for the following: | |
| Message control system (MCS) | Yes. (Only for database control.) |
| Programs | Yes |
| Output message for the following: | |
| Refresh | No. (Can be implemented by a user-created processing item.) |
| Recall | No. (Can be implemented by a user-created processing item.) |
| Placing station out of service for 15 minutes | No |
| Program in mix can be the following: | |
| Permanent in mix | Yes |
| Temporary in mix | Yes |
| Restricting commands from user-supplied program | Yes |
| Station alternates | No. (Can be implemented by a user-created processing item.) |
| Statistics | Yes |

Table A–4 compares the message formatting features of GEMCOS to COMS.

**Table A–4. Message Formatting**

| GEMCOS Feature | COMS Format Support Library Feature |
|---|---|
| Blank fill | Yes |
| Data translation [FUNCTION] | Yes |
| Forms composition | Yes |
| Input formatting by trancode | Yes |
| Item count passed to program | Yes |
| Numeric editing | Yes |
| Numeric verification and zero fill | Yes |
| On-line updating of formats | Yes |
| Output formatting by device type and output message ID | Yes |
| Paging: | |
| Input | Yes |
| Output | Yes |
| Paging breaks | Yes |
| Repeat paging | Yes |
| Total lines | Yes |
| Update | Yes |
| Variable format | Yes |
| Updated formats available in practice mode | No |
| User formatting of MCS responses | No. (Can be implemented by a user-created processing item. Note that all COMS message control system (MCS) messages go through the Menu-Assisted Resource Control (MARC) interface, and MARC has MultiLingual System (MLS) capabilities to translate the language.) |
| User-supplied formatting routines | Yes |
| Variable length of items | Yes |
| Variable repetition of items | Yes |
| Variable sequence of items | Yes |

Table A–5 compares the recovery features of GEMCOS to COMS.

**Table A–5. Recovery**

| GEMCOS Feature | COMS Feature |
|---|---|
| Archival recovery | Yes |
| Audit of input messages | Yes |
| Audit of input messages by trancode | Yes. (Auditing done by database.) |
| Audit of output messages | No. (Can be implemented by a user-created processing item.) |
| Audit of output messages by program action | No. (Can be implemented by a user-created processing item.) |
| DMSII: | |
| Recovery of program-to-program response to program messages | No |
| Recovery of program-to-program response to station messages | No |
| Synchronized recovery | Yes |
| Nonrecovery of messages | Yes |
| Recovery of output message before entering Data Management System II (DMSII) transaction state | No |
| Recovery for batch job feeding transaction programs | Yes |
| Recovery transparent to program | Yes |

Table A–6 compares the security features of GEMCOS to COMS.

**Table A–6. Security**

| GEMCOS Feature | COMS Feature |
|---|---|
| Continuous log-on | Yes |
| Dial-in stations required to identify | No |
| Identifier passed to programs [INDIVIDUALID] | Yes |
| Multiple users per station | No |
| Station able to switch between application systems | Yes |
| Station limited to trancodes for application system | Yes. (Usercode limited to windows.) |
| Usercode (no password) | Yes. (Usercode with password.) |
| User-supplied access control package | No. (Can be implemented by a user-created processing item.) |
| Valid trancodes for usercode | Yes |
| Valid usercodes for station | Yes |

Table A–7 compares the station interface of GEMCOS to COMS.

**Table A–7. Station Interface**

| GEMCOS Feature | COMS Feature |
|---|---|
| Acknowledgement of input messages as follows: | |
| LOGICALACK by message control system (MCS) | Yes. (Not on CP 2000 stations.) |
| PROGRAMACK by program | No |
| Fixed length of trancodes | Yes |
| Limit of one transaction at a time | Yes |
| Return to MCS defined by Network Definition Language II (NDLII) at end of job (EOJ). | Yes |
| Select tailored data comm software. | No |
| Specify location of trancode in message. | Yes |
| Title on output messages | No. (Can be implemented by a user-created processing item.) |

# Appendix B
# GEMCOS Sources for COMS Entities and Attributes

An entity is a category of information. Attributes describe the entity or are components of the entity. For Generalized Message Control System (GEMCOS) programs to run directly as Communications Management System (COMS) programs, GEMCOS entities must be converted into COMS entities.

The following list shows the GEMCOS counterpart for six COMS entities: agendas, programs, trancodes, device types, stations, and windows. The entities are listed as they appear in the COMS Utility Home Menu, where COMS entities are normally defined. The list shows all attributes that can be defined for that entity using the COMS Utility. If the attribute does not have a GEMCOS counterpart, a dash (–) appears in the GEMCOS column.

### Agenda Counterparts

The following entity comparison shows the GEMCOS counterpart for the COMS entities associated with agendas.

| GEMCOS Entity | COMS Entity |
|---|---|
| INPUTQUEUE <ID> + AG | Agenda name |
| SYSTEM <ID> | Window name |
| – | Default agenda |
| – | Processing-item list |
| PROGRAM <ID> | Destination |

### Program Counterparts

To alleviate migration problems and to identify the COMS program entity correctly, do the following:

- Define the USERCODE attribute if the TITLE attribute has a specified usercode.
- In the FAMILY attribute, include the family on which COMS resides.

The following entity comparison shows the GEMCOS counterpart for the COMS entities associated with programs.

| GEMCOS Entity | COMS Entity |
|---|---|
| PROGRAM <ID> | Program name |
| PROGRAM TITLE | TITLE |

*continued*

*continued*

| GEMCOS Entity | COMS Entity |
|---|---|
| PROGRAM USERCODE | USERCODE |
| PROGRAM FAMILY | FAMILY |
| PROGRAM CHARGECODE | CHARGECODE |
| PROGRAM PRIORITY | PRIORITY |
| – | Valid security-category list |
| – | Database name |
| – | Input-queue memory size |
| PROGRAM MINCOPIES | Minimum copies |
| PROGRAM MAXCOPIES | Maximum copies |
| – | Remote-file interface |
| 1st INPUTQUEUE TIMELIMIT | Time limit |
| 1st INPUTQUEUE QUEUEDEPTH | Queue depth |
| – | Remote users |

### Trancode Counterparts

The following entity comparison shows the GEMCOS counterpart for the COMS entities associated with trancodes.

| GEMCOS Entity | COMS Entity |
|---|---|
| MKE <ID> OF INPUTQUEUE | Trancode name |
| SYSTEM <ID> | Window name |
| INPUTQUEUE <ID>AG | Agenda name |
| – | Security-category name |
| MKE <ID>[<module><function>] | Module Function Index |

### Device Type Counterparts

The following entity comparison shows the GEMCOS counterpart for the COMS entity associated with a device type.

| GEMCOS Entity | COMS Entity |
|---|---|
| DEVICE <ID> | Device-type name or names |

### Station Counterparts

The following entity comparison shows the GEMCOS counterpart for the COMS entities associated with stations.

| GEMCOS Entity | COMS Entity |
|---|---|
| STATION <ID> | Station name |
| SYSTEM <ID> | Default window |
| DEVICE <ID> or STALIST | Device type |
| – | Default usercode |
| – | Valid security-category list |
| STATION MKEPOSITION | Trancode position |
| STATION SYSTEMSPOSTATION GLOBAL SYSTEMSPO | Control station |
| – | Super user |
| – | System user |
| – | Privileged user |
| STATION CONTINUOUSLOGON | Continuous log on |

### Window Counterparts

The following entity comparison shows the GEMCOS counterpart for the COMS entities associated with windows.

| GEMCOS Entity | COMS Entity |
|---|---|
| SYSTEM <ID> | Window name |
| – | Maximum users |
| – | Maximum dialogs |
| 17 | Maximum trancode size |
| – | Remote-file window |
| – | Remote-file program |
| – | Message control system (MCS) window |
| – | MCS title |
| – | Notify Open |
| – | Notification text |
| – | Notify On |
| – | Notification text |

# Glossary

## A

**agenda**
In the Communications Management System (COMS), an entity used for message routing that consists of a processing-item list and a destination. An agenda can be applied to messages that are received or sent by application programs.

**Agenda Processor library**
In the Communications Management System (COMS), an internal library that applies processing items to messages by executing the processing-item lists that agendas specify.

**ALGOL**
Algorithmic language. A structured, high-level programming language that provides the basis for the stack architecture of the Unisys A Series systems. ALGOL was the first block-structured language developed in the 1960s and served as a basis for such languages as Pascal and Ada. It is still used extensively on A Series systems, primarily for systems programming.

**attribute**
The information that describes a characteristic of an entity.

## B

**BNA**
The network architecture used on A Series, B 1000, and V Series systems as well as CP 9500 and CP 2000 communications processors to connect multiple, independent, compatible computer systems into a network for distributed processing and resource sharing.

## C

**CANDE**
*See* Command and Edit.

**COBOL(68)**
A version of the COBOL language that is compatible with the American National Standard X3.23-1968.

**COBOL74**
A version of the COBOL language that is compatible with the American National Standard X3.23-1974.

**Command and Edit (CANDE)**
A time-sharing message control system (MCS) that enables a user to create and edit files, and to develop, test, and execute programs interactively.

**Communications Management System (COMS)**
A general message control system (MCS) that controls online environments on A Series systems. COMS can support the processing of multiprogram transactions, single-station remote files, and multistation remote files. *See also* COMS (Full-Featured) and COMS (Kernel).

**COMS**
*See* Communications Management System.

**COMS (Full-Featured)**
A version of the Communications Management System (COMS) message control system (MCS) that provides full configuration capabilities through the COMS Utility. The COMS Utility enables the user to define and customize a COMS transaction processing environment, which provides additional features like transaction-based routing and database recovery. In addition, the user can track COMS statistics and use GEMCOS migration aids.

**COMS (Kernel)**
The transitional, temporary version of the Communications Management System (COMS) message control system (MCS). COMS creates a predefined configuration file that enables the user to use the window feature with the following three windows: a Menu-Assisted Resource Control (MARC) window with eight dialogs, a Command and Edit (CANDE) window with two dialogs, and a Generalized Message Control System (GEMCOS) window with one dialog. Additionally, the user can communicate with remote-file programs.

**COMS Utility**
The Communications Management System (COMS) program that defines and maintains the specifications stored in the COMS configuration file.

**conversion area**
In the Communications Management System (COMS), the user data space in the header of a message. The conversation area is user defined and can contain information passed by a program or processing item. When used with a direct-window interface, this area contains the telephone number to be dialed.

**CP 2000**
*See* CP 2000 communications processor.

**CP 2000 communications processor**
A data communications processor (DCP) that provides communications interfaces to local area networks (LANs) and wide area networks (WANs), including BNA Version 2 and Transmission Control Protocol/Internet Protocol (TCP/IP) networks. The CP 2000 also provides connections to terminals controlled by BNA Version 2 software.

# D

**Data Communications ALGOL (DCALGOL)**
A Unisys language based on ALGOL that contains extensions for writing message control system (MCS) programs and other specialized system programs.

**data communications controller (DCC)**
The subset of the master control program (MCP) operating as a group of independent tasks, each associated with one network support processor (NSP) or data communications data link processor (DCDLP).

**data communications data link processor (DCDLP)**
A data communications processor (DCP) that combines the functions of a network support processor (NSP) and a line support processor (LSP) into one physical data link processor (DLP) and supports up to four lines of communication.

**Data Management System II (DMSII)**
A specialized system software package used to describe a database and maintain the relationships among the data elements in the database.

**DCALGOL**
*See* Data Communications ALGOL.

**DCC**
*See* data communications controller.

**DCDLP**
*See* data communications data link processor.

**direct window**
In the Communications Management System (COMS), a type of window that enables the user to route messages directly to COMS, while using all the COMS capabilities for preprocessing and postprocessing of messages.

**DLS number**
A construct made up of three numbers, each separated by a colon (:). The first number is the relative network support processor (NSP) number, which was previously the data communications processor (DCP) number. The second number is the line number. The third number is the relative station number within the line.

**DMSII**
*See* Data Management System II.

# E

**end of file (EOF)**
A code at the end of a data file that signals that the last record in the file has been processed.

**end of job (EOJ)**
(1) The termination of processing of a job. (2) In the Communications Management System (COMS) and X.25, the control code that signals the receiver that a job has completed.

**end of task (EOT)**
The termination of processing of a task.

**end of transmission (EOT)**
A control code that tells the receiver that all user data (text) has been sent.

**entity**
(1) An item about which information is stored. An entity can be tangible or intangible and is further defined by attributes, which are the characteristics of the entity. (2) In the Communications Management System (COMS), a category of items within the configuration file. (3) In the Generalized Message Control System (GEMCOS), a category of information within the GEMCOS control file. (4) Any object defined in the Advanced Data Dictionary System (ADDS). To ADDS, an entity can be a Screen Design Facility (SDF) field, form, or formlibrary; an attribute or class in a Semantic Information Manager (SIM) database; a data set, group, or item in a Data Management System II (DMSII) database; or the entire SIM or DMSII database. Note that the definitions that are stored in ADDS–objects and their relationships–are themselves known as entities. (5) In the Screen Design Facility (SDF), a field, form, or formlibrary about which information is stored. (6) In the InfoExec™ environment, the basic unit of a Semantic Information Manager (SIM) database. A SIM entity can be any member of a SIM class, such as an employee, a department, or a project.

**entry point**
A procedure or function that is a library object.

**EOF**
*See* end of file.

**EOJ**
*See* end of job.

**EOT**
*See* end of task, end of transmission.

# F

**format support library (FS-LIB)**
A library provided by the Communications Management System (COMS) that is used to maintain Generalized Message Control System (GEMCOS) input formatting, output formatting, forms requests, paging, and library formats.

InfoExec is a trademark of Unisys Corporation.

**formatting**
The handling of fields within a message prior to their delivery to a program (input message) or a terminal (output message).

**FS-LIB**
*See* format support library.

# G

**GEMCOS**
*See* Generalized Message Control System.

**Generalized Message Control System (GEMCOS)**
A message control system (MCS) developed for online systems. GEMCOS is transaction oriented.

# I

**I/O**
Input/output. An operation in which the system reads data from or writes data to a file on a peripheral device such as a disk drive.

**input format**
A format for messages to be delivered to a program.

# L

**library**
(1) A collection of one or more named routines or library objects that are stored in a file and can be accessed by other programs. (2) A program that exports objects for use by user programs. (3) (VDP) A collection of related files.

**logical station number (LSN)**
A unique number assigned to each station in a network and each pseudostation allocated by a message control system (MCS). Each station has an LSN assigned according to the order in which the stations are defined.

**LSN**
*See* logical station number.

# M

**MARC**
*See* Menu-Assisted Resource Control.

**MCS**
*See* message control system.

**Menu-Assisted Resource Control (MARC)**
A menu-driven interface to A Series systems that also enables direct entry of commands.

**message control system (MCS)**
(1) A program that controls the flow of messages between terminals, application programs, and the operating system. MCS functions can include message routing, access control, audit and recovery, system management, and message formatting. (2) In X.25, a program that acts as an interface between the application and the Network Definition Language II (NDLII) and handles such functions as routing, security, auditing, and changes in the network.

**MLS**
*See* MultiLingual System.

**MultiLingual System (MLS)**
An environment that can process information using the standards and functional requirements of different localities, cultures, or lines of business. The processing of information depends on the ccsversion, languages, and conventions that are defined for the system. For example, output messages, online help text, menus, and screens can be developed and accessed in different natural languages, such as English, French, or Japanese.

# N

**NAK**
*See* negative acknowledgement.

**NDLII**
*See* Network Definition Language II.

**negative acknowledgment (NAK)**
A response that occurs if a terminal is not able to respond to a poll/select inquiry.

**Network Definition Language II (NDLII)**
The Unisys language used to describe the physical, logical, and functional characteristics of the data communications subsystem to network support processors (NSPs), line support processors (LSPs), and data communications data link processors (DCDLPs).

**network support processor (NSP)**
A data communications subsystem processor that controls the interface between a host system and the data communications peripherals. The NSP executes the code generated by the Network Definition Language II (NDLII) compiler for line control and editor procedures. An NSP can also control line support processors (LSPs).

**NSP**
*See* network support processor.

# O

**ODT**

*See* operator display terminal.

**operating system**

The set of programs that control the operational environment of a computer system by activities such as managing processors, memory, and peripherals; logging system activities; enforcing security; and executing system commands. On A Series systems, the operating system consists of a master control program (MCP) and system libraries such as CENTRALSUPPORT, GENERALSUPPORT, JOBFORMATTER, and PRINTSUPPORT.

**operator display terminal (ODT)**

A terminal or other device that is connected to the system in such a way that it can communicate directly with the operating system. The ODT allows operations personnel to accomplish system operations functions through either of two operating modes: system command mode or data comm mode.

**output format**

A format for messages to be delivered to a terminal.

# P

**paged format**

A format for a message that cannot be displayed on a terminal in one screen, or for a message that is specifically divided into several screens.

**postprocessing**

The processing done to a message by processing items after an application program sends out the message.

**preprocessing**

The processing that the Agenda Processor performs on a message before an application program receives the message.

**processing-item library**

In the Communications Management System (COMS), a user-written ALGOL library containing a set of procedures called processing items. A processing-item library can be called only by the COMS Agenda Processor library to preprocess and postprocess messages as they are received and sent by programs.

# R

**relative station number (RSN)**

An integer value identifying one station of a multistation remote file. The value is contained in the LASTSUBFILE attribute for the file. This value is used to determine the station read from or written to when using the READ or RECV operation, or the SEND or EXCPT operation, respectively.

**remote file**
A file with the KIND attribute specified as REMOTE. A remote file enables object programs to communicate interactively with a terminal.

**Remote Print System (ReprintS)**
A Unisys software system that controls the routing and printing of backup files at remote (data comm) destinations and on BNA networks.

**ReprintS**
*See* Remote Print System.

**restart data set (RDS)**
In Data Management System II (DMSII), a data set containing restart records that application programs can access to recover database information after a system failure.

**RSN**
*See* relative station number.

# S

**Screen Design Facility (SDF)**
The InterPro product used for creating forms for online, transaction-based application systems.

**SDF**
*See* Screen Design Facility.

**synchronized recovery**
(1) In the Communications Management System (COMS), a function that resubmits incomplete transactions to the database after a transaction-state abort, system crash, or rollback occurs. This COMS function is called synchronized recovery because it reprocesses transactions in the same order that they were originally processed by multiple programs running asynchronously, even if the transactions were conflicting. (2) In the Generalized Message Control System (GEMCOS), recovery of transactions running concurrently prior to a system or program failure in the original database update sequence. This recovery is invisible to application programs and terminal operators.

# T

**TCL**
*See* transaction control language.

**trancode**
*See* transaction code.

**transaction code (trancode)**
(1) A sequence of characters included in a message that indicates the agenda to apply to a message during preprocessing or postprocessing. (2) In the Communications Management System (COMS), a code that can appear in a transaction-initiating message

header, indicating the processing that is to be carried out. This code is used to route the message to the appropriate host program.

**transaction control language (TCL)**
A language that is used to describe stations, programs, and message control system (MCS) capabilities.

**transaction state**
In Data Management System II (DMSII), the period in a user-language program between a begin transaction operation and an end transaction operation.

# W

**window**
(1) In the Communications Management System (COMS) architecture, the concept that enables a number of program environments to be operated independently and simultaneously at one station. One of the program environments can be viewed while the others continue to operate. (2) In the Editor, a conceptual aperture through which any screen-sized group of lines in a work file can be viewed. The window is said to move forward in the file if lines of higher sequence numbers are being displayed, and backward if lines of lower sequence numbers are being displayed. (3) In the Forms Manager, a rectangular area in a form. (4) In data communications, a flow control mechanism, the size of which is equal to the number of frames, packets, or messages that can be sent from a transmitter to a receiver before any reverse acknowledgment is required. At the data terminal equipment (DTE)/data circuit terminating equipment (DCE) interface of a logical channel, a window is the maximum number of consecutive packets that can be transmitted. (5) A portion of a screen that has been allocated to display the contents of a specified area of memory.

# Bibliography


*A Series ALGOL Programming Reference Manual, Volume 1: Basic Implementation* (form 8600 0098). Unisys Corporation.

*A Series ALGOL Programming Reference Manual, Volume 2: Product Interfaces* (form 8600 0734). Unisys Corporation.

*A Series COBOL ANSI-74 Programming Reference Manual, Volume 1: Basic Implementation* (form 8600 0296). Unisys Corporation.

*A Series COBOL ANSI-74 Programming Reference Manual, Volume 2: Product Interfaces* (form 8600 0130). Unisys Corporation.

*A Series Communications Management System (COMS) Capabilities Manual* (form 8600 0627). Unisys Corporation.

*A Series Communications Management System (COMS) Configuration Guide* (form 8600 0312). Unisys Corporation.

*A Series Communications Management System (COMS) Operations Guide* (form 8600 0833). Unisys Corporation.

*A Series Communications Management System (COMS) Programming Guide* (form 8600 0650). Unisys Corporation.

*A Series DCALGOL Programming Reference Manual* (form 8600 0841). Unisys Corporation.

*A Series Generalized Message Control System (GEMCOS) Operations Guide* (form 8600 0676). Unisys Corporation.

*A Series I/O Subsystem Programming Guide* (form 8600 0056). Unisys Corporation. Formerly *A Series I/O Subsystem Programming Reference Manual* .

*A Series Menu-Assisted Resource Control (MARC) Operations Guide* (form 8600 0403). Unisys Corporation.

*A Series Mark 3.9 Software Release Capabilities Overview* (form 8600 0015). Unisys Corporation.

*A Series Print System (PrintS/ReprintS) Administration, Operations, and Programming Guide* (form 8600 1039). Unisys Corporation.

*A Series System Commands Operations Reference Manual* (form 8600 0395). Unisys Corporation.

*A Series System Software Support Reference Manual* (form 8600 0478). Unisys Corporation.

# Index

## A



## C



## D

## E



## F



## G



## I



## K



## L

## M



## P



## R



## S



## T

## U



## W

UNISYS

# Help Us To Help You

Publication Title

Form Number

Unisys Corporation is interested in your comments and suggestions regarding this manual. We will use them to improve the quality of your Product Information. Please check type of suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

Name Telephone number ( )

Title Company

Address

City State Zip code

Cut along dotted line

Tape

Please Do Not Staple

Tape

Fold Here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS MAIL PERMIT NO. 817 DETROIT, MI

POSTAGE WILL BE PAID BY ADDRESSEE

**UNISYS CORPORATION**
**ATTN: PUBLICATIONS**
**25725 JERONIMO ROAD**
**MISSION VIEJO, CA 92691-9826**

86001567-000