INSTRUCTIONS

Because all our documentation is being reproduced on reduction, it is essential that this form is either TYPED OR PRINTED LEGIBILY IN BLACK INK.

Fill out only those fields marked by bullets (o).

DOCUMENT TYPE: One of the following:
- AR - Archictecture Requirments (CYBER 80)
- ESL - Baseline Documentation Change(s)
- DAP - Design Action Paper
- DDD - Design Direction Document
- DR - Design Requirements
- EPP - Evalutaion Project Plan
- ERS - External Reference Specifications
- FTP - Feature Test Plan
- GDS - General Design Spec (CYBER 80)
- GID - General Internal Specifications
- IHB - Installation Handbook
- IPP - Implementation Project Plan
- PDA - Product Development Assurance Plan
- PJP - Project Plan
- SID - Standards Document
- STP - System Test Plan
- TIR - Transmittal, Integration & Release Criteria
- OTH - Other, specify type

TITLE: Be concise. Be sure to include the name of the product involved.

ABSTRACT: A brief description of the contents. If the item is extremely short, it may be written in designated space with nothing else attached.

PUBLICATION #: If a document with a pub. # is being changed, write the pub. # here.

RSM & PRM Numbers: If any are associated with this document, list them here.

APPROVALS: (Minimum DCS Requirements)
Project leader for all.
Unit Manager for all.
Section Manager: Project Plans only.
DCS will assign referee.

PROJECT: The project name that the author belongs to, e.g., CLUE.

DCS Mailing List Pasted Here -
(On top of instructions)

---

*For ARHOPS Use Only:

CONTROL DATA CORPORATION

DOCUMENT CONTROL FORM

o DOCUMENT TYPE: DAP

o LOG ID: ARH7345

o TITLE: GUIDELINES FOR THE DEFINE/START/CHANGE/STOP/CANCEL COMMANDS

o ABSTRACT: PROVIDES GUIDELINES FOR THE IMPLEMENTATION OF CDCNET COMMANDS

o PRODUCT AFFECTED: CDCNET

o AUTHOR: B. S. SEKHON

o MAIL STATION: ARH207 o EXTENSION: 3367

o PROJECT: CDCNET DESIGN

o SECTION: ACSD

o RSM or PSR #:

o SIP Number(s):

o PUB #:

o PRM #: o Redesign* o Reimplementation

	Name	Approval Initials	Date
o Internal Reviewer*			
o Project Leader			
o Unit Manager	R. C. Anderson	RCA	5/30/86
o Section Manager			
Design Team Referee	J. D. REED	JDR	5/29/86

DESIGN TEAM: CDCNET DEFAULT CYCLE: C WORKING DAYS

o Special Distribution:

Referee's Distribution Codes: AC-D

NUMBER	DATE	TYPE	SUBMITTED BY	DEFAULT APPROVAL DATE	CURRENT STATUS: REVIEW	HOLD	APPROVED	WITHDRAWN
1	6/02/86	Orig	B. S. Sekhon	NONE			A	

79 IM - 0145H/0146H

ARH7345

86/05/29

GUIDELINES FOR THE DEFINE/START/CHANGE/STOP/CANCEL COMMANDS

B. S. Sekhon
05/29/86

DISTRIBUTION LIST PAGE 1

ARH7345

. I	SVL107	R.A.MANN	. I	ARH207	A.A.WEBER
. I	SVL107	R. WESTGAARD	. I	ARH207	A.C.RUPERT
. I	SVL107	P.A.SHIREMAN	. I	ARH207	A.R.FIEDLER
. I	SVL107	P.L.MCNAIR	. I	ARH207	G.S.JUNCKER
. I	SVL107	P.L.KENNY	. I	ARH207	T.C.MCGEE
. I	SVL107	K.M.MIYAHARA	. I	ARH207	M.E.MADDEN
. I	SVL107	A.E.ANDERSON	. I	ARH207	K.G.VIGGERS
. I	SVL107	R.M.FARRELL	. I	ARH207	R.S.BROWN
. I	SVL110	C. CHOW	. I	ARH207	T.E.BERETT
. I	SVL126	P.J.TRETTEL	. I	ARH207	R.E.ERICKSON
. I	SVL161	R.W.FLETCHER	. I	ARH207	K.D.DEHNERT
. I	SVL161	E.P.LAMOUREUX	. I	ARH207	R.L.DELANEY
. I	SVL170	J.C.LEE	. I	ARH207	R.C.ANDERSON
. I	ARH201	D.K.TAYLOR	. I	ARH207	R. WOODRUFF
. I	ARH207	B.T.ZEMLIN	. I	ARH207	L.A.BERKEBILE
. I	ARH207	J.P.YOUNG	. I	ARH207	J.K.MATTHEWS
. I	ARH207	R.R.RUNDQUIST	. I	ARH207	M.J.AMUNDSON
. I	ARH207	R.E.TATE	. I	ARH213	C.J.DUFFY
. I	ARH207	H.G.COVERSTON	. I	ARH213	J.D.KNODEL
. I	ARH207	D.A.NETLAND	. I	ARH219	J.T.LINDMEYER
. I	ARH207	J.D.REED	. I	ARH219	D.W.GROUT
. I	ARH207	S.D.BECCHETTI	. I	ARH242	R.A.JAPS
. I	ARH207	B.S.SEKHON	. I	ARH248	D.K.ELDRED
. I	ARH207	A.T.STAGG	. I	ARH254	R.D.SWAN
. I	ARH207	J.A.WACHTER	. I	ARH254	J.E.ESLER
. I	ARH207	S.M.KEEFER	. I	ARH263	J.L.NADING
. I	ARH207	J.A.DAYKIN	. I	ARH293	N.L.REDDY
. I	ARH207	E.E.NELSON	. I	AHS279	J.R.HILDEBRAND
. I	ARH207	L.L.LUCAS	. I	HQC02J	T.J.ROCHE
. I	ARH207	R.L.SUNDBERG	. I	PLY011	R.L.CLARK
. I	ARH207	S.L.HODGES	. I	RVL004	D.A.SEIDENSTRICKER
. I	ARH207	F.A.ARNESON	. I	STAOPS	M.C.STEELE
. I	ARH207	G.A.KERSTEN	. I	STAOPS	J.F.MEYER

REVISION RECORD			
Revision	Description	Author	Date
A	TDRB draft	BSS	03/11/86
B	Resolution of TDRB comments	BSS	05/16/86
C	Resolution of pre-DCS comments	BSS	05/29/86

RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

Table of Contents

1.0 INTRODUCTION 1-1

2.0 SITUATION REQUIRING ACTION 2-1
2.1 BACKGROUND 2-1
2.1.1 CDCNET ENTITIES 2-1
2.1.2 MANAGEMENT OF CDCNET SERVICES 2-1
2.1.3 SERVICE INTERFACES 2-2
2.2 ISSUES AND CONCERNS 2-2

3.0 REQUIREMENTS AND CONSTRAINTS 3-1
3.1 REQUIREMENTS 3-1
3.2 CONSTRAINTS 3-1

4.0 RECOMMENDATION 4-1
4.1 CDCNET SERVICES/ENTITIES WHICH NEED TO BE EXTERNALIZED 4-1
4.2 SERVICE INTERFACES TO ACCESS AND MANAGE SERVICES WITHIN A SYSTEM 4-3
4.3 GUIDELINES TO GROUP INDIVIDUAL ENTITIES 4-4
4.4 GUIDELINES TO DETERMINE IF A COMMAND IS NEEDED . . 4-7
4.4.1 DEFINE COMMAND 4-7
4.4.2 START COMMAND 4-8
4.4.3 CHANGE COMMAND 4-8
4.4.4 STOP COMMAND 4-9
4.4.5 CANCEL COMMAND 4-9
4.5 IMPLEMENTATION GUIDELINES 4-10
4.5.1 IMPLEMENTATION GUIDELINES FOR SERVICE INTERFACES 4-10
4.5.1.1 INTERFACE DATA STRUCTURE 4-11
4.5.1.2 COMMAND PROCESSOR SUB-INTERFACE 4-12
4.5.1.3 SERVICE USER SUB-INTERFACE 4-12
4.5.1.4 SERVICE PROVIDER SUB-INTERFACE 4-13
4.5.2 DEFINE COMMAND 4-13
4.5.3 START COMMAND 4-15
4.5.4 CHANGE COMMAND 4-17
4.5.5 STOP COMMAND 4-17
4.5.6 CANCEL COMMAND 4-18
4.6 MISCELLANEOUS RECOMMENDATIONS 4-19
4.6.1 LOADING OF SOFTWARE TO SUPPORT ENTITIES 4-19
4.6.2 ORDER OF COMMAND EXECUTION 4-20
4.6.3 GENERATION OF COMMAND RESPONSE 4-20
4.7 IMPLEMENTATION RECOMMENDATIONS 4-21

5.0 SUPPORTING EVALUATION 5-1
5.1 COMPATIBILITY IMPACT 5-1
5.2 ALTERNATE SOLUTIONS/APPROACHES 5-2

6.0 REQUIRED ACTIONS 6-1

APPENDIX A - COPY OF RE-CONFIGURATION DAP A-1

RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

A1.0 INTRODUCTION A1-1

A2.0 SITUATION REQUIRING ACTION A2-1
A2.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM A2-1
A2.2 CONFIGURATION COMMAND RESPONSE SUPPORT A2-1

A3.0 CONSTRAINTS AND REQUIREMENTS A3-1

A4.0 RECOMMENDED TECHNICAL SOLUTION A4-1
A4.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM A4-1
A4.2 CONFIGURATION COMMAND RESPONSE SUPPORT A4-2

A5.0 SUPPORTING EVALUATION A5-1
A5.1 ALTERNATE SOLUTIONS/APPROACHES A5-1
A5.1.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM A5-1
A5.1.2 CONFIGURATION COMMAND RESPONSE SUPPORT A5-1

A6.0 REQUIRED ACTIONS A6-1

1.0 INTRODUCTION

1.0 INTRODUCTION

During design and development of new CDCNET features, questions are often raised about commands needed to support them. Invariably, questions also come up about implementation choices for the associated command processors. One obvious reason for these questions is the lack of guidelines for the definition and implementation of commands. Specifically, guidelines are needed determine which commands are needed to support a new feature. Guidelines are also needed to determine what functions are provided by a given command and how it must be implemented. This design note provides these guidelines.

The term "commands" as used in the rest of this document refers to the define, start, change, stop and cancel commands. These commands are also referred to as the service management commands. This design note does not address commands provided by the terminal user interface software.

This document uses the terms entity and service interchangeably. In general, there is a one to one relationship between an entity and a service. For example, the File access M-E entity provides a service which allows CDCNET software components to access network files. However, in some cases the relationship between an entity and a service is not very explicit. For example, the HDLC network solution entity does provide a service which allows two CDCNET systems to be connected across an HDLC link. However, this service is not as visible as the one provide by the File access M-E.

The guidelines provided in this design note will help in identifying the need for commands to support new features. These will also help in achieving consistent design and implementation of command processors and interfaces needed to support CDCNET commands.

Section 2 of this design note describes issues and concerns which must be taken into account by the proposed guidelines. These concerns are consolidated into requirements in section 3. Section 4 provides recommended guidelines for different aspects of CDCNET commands.

2.0 SITUATION REQUIRING ACTION

2.0 SITUATION REQUIRING ACTION

2.1 BACKGROUND

2.1.1 CDCNET ENTITIES

Since CDCNET is based on the OSI reference model, most of its software components (also called entities) provide services and functions which are associated with a single layer in the OSI model, a management function, a gateway or an interface to a non-CDNA system. The following are some examples of CDCNET entities.

- X.25 Gateway
- X.25 Packet level
- BIP
- SVM
- MCI Driver
- MCI SSR
- XNS Transport
- Async TIP
- Async line connected to a CDCNET system
- File Access Support

Each CDCNET system contains a large number of entities. Each entity provides a specific set of services. However, users of a CDCNET system as well as people responsible to manage the CDCNET network, in general, will not be interested in services provided by individual CDCNET entities. Instead, they will be interested in services provided by a group of entities working as a single unit. For example, a network administrator will be more interested in the HDLC network solution as a single service rather than individual layered entities which provide layers 1, 2 and 3A functions for the HDLC network solution.

2.1.2 MANAGEMENT OF CDCNET SERVICES

The following aspects of CDCNET services need to be managed.

2.0 SITUATION REQUIRING ACTION
2.1.2 MANAGEMENT OF CDCNET SERVICES

- Since some services may not be present in each CDCNET system at all times, a method is needed to add or delete them from a system.

- Some services may not be logically active at all times. Also, it may be necessary to control the availability of certain services. Therefore a method is needed to activate and deactivate services as needed.

- Some attributes of certain services may be configurable. Therefore, a method is needed to specify and/or change attributes of such services.

CDCNET supports management of its services via commands. Define, start, change, stop and cancel commands are used to provide the needed management functions for a given service. In the rest of this document, these commands are referred to as the service management commands.

2.1.3 SERVICE INTERFACES

Each CDCNET entity which provides a service, needs to provide an interface which may be used by other entities to obtain its services. Since each CDCNET system includes a large number of entities, a large number of such interfaces also exist or need to be defined. However, a lot of these interfaces are similar to each other and can potentially be replaced with (or provided by) a small number of common service interfaces.

2.2 ISSUES AND CONCERNS

In view of the above background, the following issues need to be addressed.

1. Even though each CDCNET system includes a large number of entities which provide services, only services provided by a small sub-set of these entities need to be externalized. Therefore it is necessary to identify the CDCNET services which are to be externalized.

   The term externalized as used here means making the

service or associated entity known outside of the CDCNET system and providing one or more service management commands for it.

2. Common service interfaces and associated entities whose services may be obtained via these interfaces need to be identified.

3. Guidelines are needed to determine how and which CDCNET entities may be combined to provide the services which are to be externalized.

4. Guidelines are needed to determine which service management commands are needed for each externally visible service and the associated entity or group of entities.

5. A definition of functions provided by command processors for service management commands as well as guidelines for their implementation are needed.

In addition to the above issues, the following secondary issues and concerns about the service management commands also need to be addressed.

A. Guidelines are needed for the use of the start parameter on the define command. At present some define commands include a start parameter. The issue which needs to be addressed is should all define commands support a start parameter. Also, should the same default value be used for this parameter in all define commands, or should it depend on the entity being defined.

B. Are there any requirements on the order in which commands for related entities must be executed. For example, can a network solution be defined (via the define command) before defining the underlying trunk. Similarly are there any restrictions on the order in which the X.25 gateway and the X.25 packet level may be started via the start commands.

C. If related entities can be started in any order, then we need a common way to deal with the situation when an entity which has been started can not provide the service it is supposed to provide, because a related entity, whose services it requires, has not been started, or has failed.

D. When should the software for an entity be loaded. The obvious choices are: when it is defined, when it is

2.0 SITUATION REQUIRING ACTION
2.2 ISSUES AND CONCERNS

started and when it is first used.

E. Should the command processors for start commands also include code to complete the initialization (e.g. opening of a SAP) for the software being started by the command processor.

F. Should the command processor for start commands automatically start the underlying software entities which are required by the software being started by the command processor.

G. Guidelines are needed to determine when a command response should be generated. Possible options are (1) when the command has been received and parsed, (2) the service to be provided by the command has been enabled, (i.e., the service is not available yet, however, everything needed to make the service available has been completed). or (3) the service to be provided by the command has become available.

3.0 REQUIREMENTS AND CONSTRAINTS

3.0 REQUIREMENTS AND CONSTRAINTS

3.1 REQUIREMENTS

The following requirements for the management of services provided by CDCNET to the network manager and/or operator need to be addressed.

1. It must be possible to add an optional service to a CDCNET system. One must be able to do so with or without enabling the service at the time it is added to the system.

2. It must be possible to delete an optional service from a CDCNET system.

3. It must be possible to initially define or set the configurable attributes of a CDCNET service and change the values of these attributes if necessary.

4. It must be possible to enable or disable an optional service at any time.

3.2 CONSTRAINTS

The solution to meet above requirements must take into account the following constraints.

1. Enabling or disabling of a service in a CDCNET system must not require any knowledge about how it is being provided by the system.

2. The method used to meet the requirements must be simple and consistent for all services.

4.0 RECOMMENDATION

4.0 RECOMMENDATION

This section provides the recommended solution for the issues identified in section 2.2. The first five sub-sections provide solution for the five major issues described in section 2.2.

A concept of entities being at different levels, i.e. higher or lower with respect to each other is used in the description of the recommended solution. An entity is considered to be at a higher level with respect to another entity, if it uses the services provided by the other entity. Similarly an entity is said to be at a lower level with respect to another entity, if it provides a service to the other entity.

4.1 CDCNET SERVICES/ENTITIES WHICH NEED TO BE EXTERNALIZED

The CDCNET services or entities which must be "externalized" (see section 2.2 for a definition of this term) can be grouped into a small number of groups. The following table shows different groups and services/entities included in each group.

Please note that the term group as used here has nothing to do with the concept of a "group" entity which is introduced in section 4.3. In order to avoid any confusion, this section will use the term "type" instead of "group".

Thus, entities belonging to the same group will be said to be of the same type.

TYPE	SERVICES/ENTITIES
MANAGEMENT FUNCTION	Operator Support File Access support Log support Alarm support Clock support Remote load support
INTERFACE	C170 Network products Interface

4.0 RECOMMENDATION
4.1 CDCNET SERVICES/ENTITIES WHICH NEED TO BE EXTERNALIZED

	X.25 Interface
GATEWAY	X.25 Gateway C170 Network products Batch Gateway C170 Network products IVT Gateway C170 Network products A-A Gateway
NETWORK SOLUTION	HDLC Ethernet Channelnet X.25 network solution
LAYER ENTITIES	CDNA Session Layer CDNA Generic Transport XNs Sequence Packet Transport XNS Internet
TERMINAL SUPPORT	
TIP	Async TIP X.25 Async TIP Hasp TIP URI TIP
LINE	Asynchronous line Synchronous line
DEVICE	terminal Device Batch Device NP Batch Device
STATION	I/O Station User I/O Station

Section 4.3 provides guidelines to determine how one or more CDCNET entities may be grouped to provide the services listed in the above table. Therefore, information in this section and in section 4.3 compliments each other.

4.2 SERVICE INTERFACES TO ACCESS AND MANAGE SERVICES WITHIN A SY EM

Each CDCNET system must include a set of common Service interfaces to obtain and manage the services provided by CDCNET entities within a CDCNET system. All CDCNET entities must offer their services through one of these interfaces.

Even though the services provided by each common Service interface will have unique characteristics, the following general capabilities must be supported by all interfaces.

1. Ability to determine if a service or entity is defined.

2. Ability to determine the status of the service or entity.

3. Ability to establish an association with the Service Interface for the purpose of obtaining the services accessible through the Service interface. (e.g. provide a capability to open a SAP).

4. Ability to delete the association with the Service Interface.

5. Automatically notify users about changes in the status of the service or entity to those users who have established an association with it.

6. Provide an interface to the service management command processors to facilitate the definition and activation of the services accessible through the common Service interface.

Initially CDCNET must provide the following Service interfaces.

A. Application and Gateway Service interface

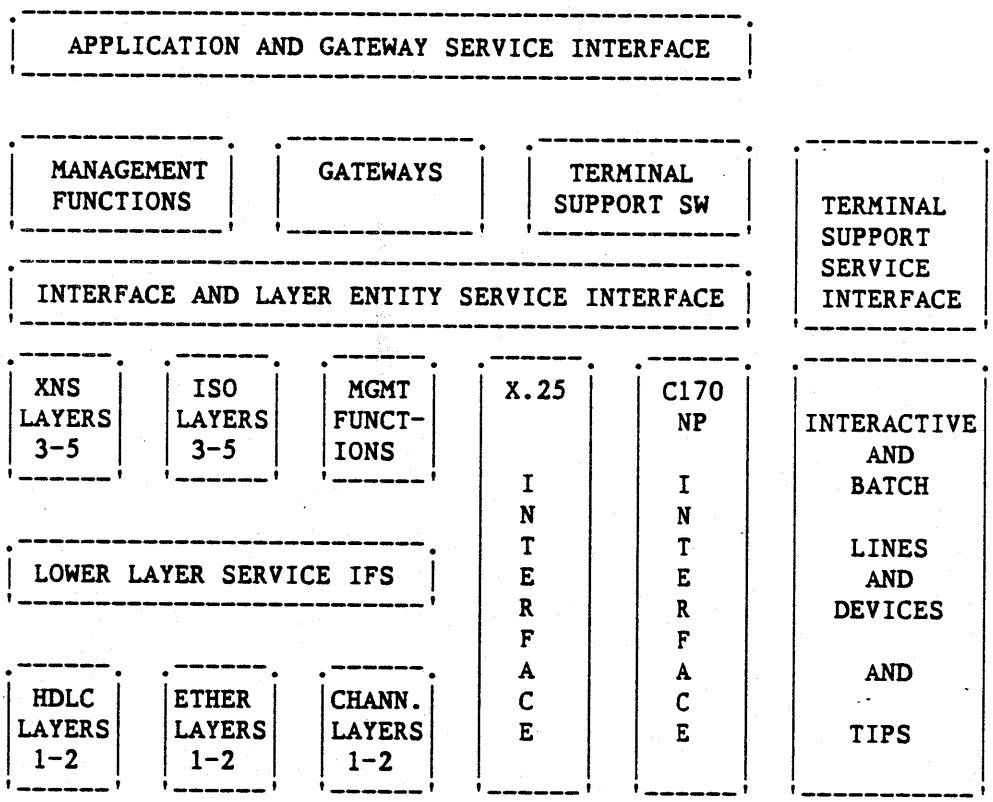
B. Interface and Layer entity Service interface

C. Lower Layer Service interface

D. Terminal Support Service interface

4.0 RECOMMENDATION
4.2 SERVICE INTERFACES TO ACCESS AND MANAGE SERVICES WITHIN A SYSTEM

The following diagram shows the group of entities which will be served by these interfaces.



The Terminal support service interface is described in the Terminal support ERS. The Lower layer Service interface is described in the Generic 3A ERS. The Application and Gateway as well as the interface and Layer entity Service interfaces have not been formally documented, and will be addressed in a separate DAP. Section 4.5.1 of this DAP provides general implementation guidelines for a Service interface. It must be noted that a detailed description of these interfaces is outside the scope of this DAP.

4.3 GUIDELINES TO GROUP INDIVIDUAL ENTITIES

Since CDCNET is based on the OSI reference model, its implementation has preserved layer boundaries. Therefore,

most of the CDCNET software components (also referred to as entities) provide services and functions which are associated with a single layer in the OSI model. For example, HDLC SSR provides layer 2 functions for a HDLC network solution (sub-network in ISO/OSI terminology). Thus services provided by CDCNET could be viewed as a collection of services provided by individual layer entities.

However, services provided by an individual layer entity may not be very interesting or useful to persons who need to manage or use CDCNET. From their perspective, a collection of services provided by more than one layer entity may together form a more meaningful service. For example, a network administrator may be more interested in managing HDLC network solution as a single service rather than individual layered entities which provide layer 1, 2 and 3A functions for the HDLC network solution.

Therefore, guidelines are needed to determine how individual entities may be combined so that they can be managed as a group and the services provided by them as a group can be externalized. The following guidelines are provided to determine if a set of entities may be combined into what is called a group entity. Each guideline is illustrated via examples.

1. All entities in the set together provide a service which is visible outside the CDCNET system and needs to be managed as a single service.

   For example, MCI Driver, MCI SSR, BIP and SVM together provide an interface to C170 network products. Since, a CDCNET system can be connected to more than one C170 mainframe, one can justify the need to manage the C170 network products interface as a single service. This will justify combining the MCI Driver, MCI SSR, BIP and SVM into a single group entity called the C170 network products interface. On the other hand, MCI Driver and MCI SSR do not provide a meaningful service by themselves and therefore may not be combined to form a group entity.

2. An entity must not be included in more than one group entity except in the case where an independent method exists to manage its definition and use within each group entity in which it is included.

   For example, the MCI SSR may be included in both the Channelnet network solution and the C170 network products

interface. This is okay because a separate LIB data structure is used for its definition and it exists as separate tasks for its use within the channelnet network solution and the C170 network products interface.

3. This guideline supplements the previous guideline. It requires that the services provided by a subset of the entities which does not include the entity at the highest level must not be made available to more than one user at the same time.

   As an example, this guideline will prevent creation of a group entity which included X.25 gateway, X.25 packet level and HDLC SSR because in this case the X.25 packet level and HDLC SSR will provided services to both the X.25 gateway as well as X.25 PAD TIP. To illustrate further the terms used in

   To further illustrate the terms used in this guideline, in the above example, X.25 gateway will be the entity at the highest level. X.25 packet level and HDLC SSR will form the subset of the entities which does not include the entity at the highest level.

The following are examples of legal group entities which may be defined using above guidelines.

- HDLC network solution which includes HDLC portion of 3A, HDLC SSR and CIM firmware for HDLC support.
- Channelnet network solution which includes MCI portion of 3A, MCI SSR and MCI driver.
- C170 network products interface which includes BIP, SVM, MCI SSR and MCI driver.
- X.25 interface which includes X.25 packet level, HDLC SSR and CIM firmware for HDLC support.
- ISO middle layer group which includes ISO session, transport and network layers.

The following are examples of group entities which do not follow the above guidelines and therefore will be illegal.

- A group entity which included Independent File Access

M-E, BIP, SVM, MCI SSR and MCI Driver.

- A group entity which included X.25 Gateway, X.25 Packet level, HDLC SSR SSR and HDLC portion of the CIM controlware.

- A group entity which included Internet layer, Intranet layer, ESCI SSR and ESCI Driver.

Each group entity must be treated as a single entity as far as the guidelines being recommended in this DAP are concerned. There is no requirement for individual entities which make up a group entity to follow these guidelines.

4.4 GUIDELINES TO DETERMINE IF A COMMAND IS NEEDED

This section provides guidelines to determine if an individual command is needed for a service, entity or a group entity. The terms "service", "entity" and "group entity" are used interchangeably in this document.

4.4.1 DEFINE COMMAND

A DEFINE command will be needed for an entity if one or more of the following guidelines are applicable to it.

1. The entity is an optional entity (i.e. not automatically present in each CDCNET system) and is not included in a group entity. Some examples of optional entities are X.25 gateway, Independent Clock M-E. X.25 interface and C170 network products interface.

2. The entity is not an optional entity and supports one or more configurable options which must be specified. In other words, the entity supports a configurable option for which a default does not exist. At present, CDCNET system does not include any such entity. This guideline is included for future considerations. (A catenet wide entity called boot defaults may be used as an example to illustrate this guideline. This entity has a configurable option called default_version which must be specified).

4.4.2 START COMMAND

A START command will be needed for an entity only if all of the following conditions are true.

1. The entity provides an optional service. In other words the service provided by the entity is not required to be available in each CDCNET system at all times.

2. The entity does not belong to a group where it can or does get started as a result of starting the group entity.

3. The entity does not provide a management function. Some examples of management functions are community support, log message transmission and file access support.

4.4.3 CHANGE COMMAND

It is difficult to provide crisp guidelines to determine if a change command is required for an entity. The reason for this is that the function provided by a change command can be obtained most of the time by first cancelling the entity and then defining it with new values for its configurable attributes.

However, it may not be always possible or desirable to cancel an entity in order to change some of its attributes. For example, consider the commands used to manage source log groups. The DEFSLG command allows one to specify a list of log messages whose transmission is enabled. The CHASLG command allows one to add or delete log messages from this list. Due to the limitations on the maximum size of a command, one may not be able to use the DEFSLG command to provide the complete list of log messages. In this case, one will have to use the CHASLG command to add remaining log messages to the list of enabled log messages.

Similarly, it may not be possible to cancel an entity if it is being used. Therefore a change command will be required for certain entities. The following guidelines must be used to determine the need of a change command.

1. A change command will be required for an entity if it supports configurable attributes and is not explicitly defined via a define command.

4.0 RECOMMENDATION
4.4.3 CHANGE COMMAND

2. A change command will be required for an entity if the break in the service caused by cancelling and redefining it with new values for its attributes via the define command is not be acceptable or desirable. Since the terms acceptable and desirable are subjective, the following examples are provided to illustrate this guideline.

   For example consider a network solution. The cost of a network solution is one of its configurable attributes. The use of cancel and define commands to change the cost of a network solution will result in major disruption and therefore will not be desirable.

   Use of a define command to accomplish a change will require reentering even those parameters which are not to be changed. This will not be desirable if the define command supports a lot of parameters.

3. A change command will be required for an entity if the limitation on the maximum size of a command will prevent one to specify values of all configurable attributes via the define command. In such cases, the change command will be required to supplement the define command.

4.4.4 STOP COMMAND

The "start" and "stop" commands will always exist as a pair. Therefore, a stop command will be needed for an entity if it needs a start command.

4.4.5 CANCEL COMMAND

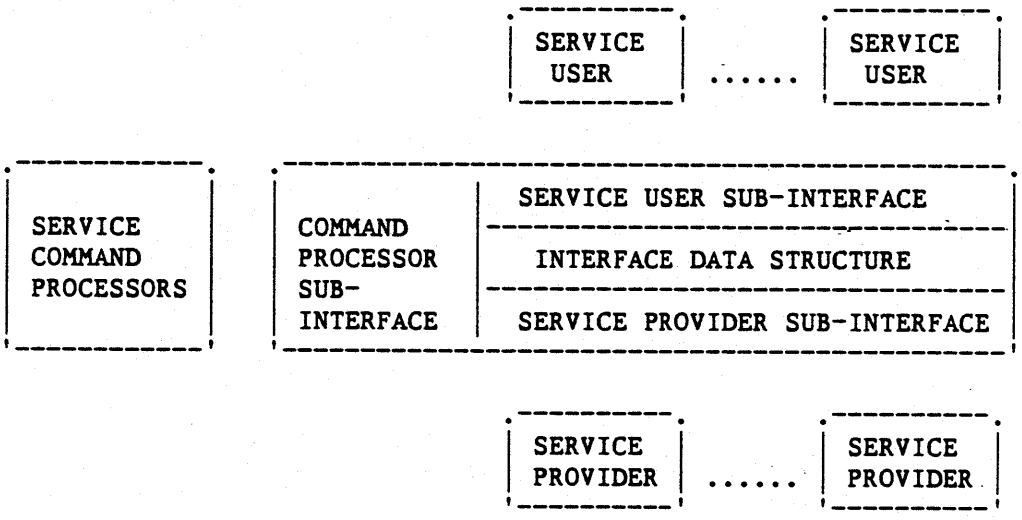
The "define" and "cancel" commands will always exist as a pair. Therefore, a cancel command will be needed for an entity if it needs a define command.

4.5 IMPLEMENTATION GUIDELINES

This section provides guidelines for the implementation of service management command processors as well as the Service interfaces.

4.5.1 IMPLEMENTATION GUIDELINES FOR SERVICE INTERFACES

The following diagram shows a general model which may be used to implement a service interface. Since services provided by each service interface will be unique, exact conformance to this model will not be required.



Each Service interface must include three sub-interfaces. One sub-interface, called the command processor sub-interface, will provide common services to the service management command processors for the services/entities accessible through the Service interface.

The second sub-interface, called the service user sub-interface, will allow users of the services to obtain the desired service. The third sub-interface, called the service provider sub-interface, will provide an interface between the

4.0 RECOMMENDATION
4.5.1 IMPLEMENTATION GUIDELINES FOR SERVICE INTERFACES

entities which provide services and rest of the Service interface software. In addition, each Service interface will include a data structure called Interface Data Structure (IDS) which will be used to maintain information about entities whose services are being provided through the Service interface.

Implementation guidelines for the interface data structure and each sub-interface are described next.

4.5.1.1 INTERFACE DATA STRUCTURE

Each Service interface must provide a method to identify the entities whose services are accessible through the interface. This method must use a two part identifier for an entity. The first part must be a name whose size must not exceed 31 characters. A portion of this name may include a fixed or constant string. The second part of the identifier must specify the type (e.g. management function, network solution) of the entity. Section 4.1 includes a list of valid entity types.

Each Service interface must maintain information about entities it serves in an interface specific data structure. This data structure must include a separate entry for each entity known to the interface. Such an entry must be created when the entity is first defined and must be retained till the entity is cancelled. The interface specific data structures will be referred to as IDSs in the rest of this document. Each entry in the IDS must include at least the following information.

- Identification of the entity (i.e. its name and type)
- Status of the entity (i.e. defined, enabled or active)
- Pointer to an entity specific definition record which contains the values of configurable attributes of the entity.

ASSUME "entities it serves" means the module/service whose IDS this is, NOT users of the module/service of this IDS.

4.0 RECOMMENDATION
4.5.1.2 COMMAND PROCESSOR SUB-INTERFACE

4.5.1.2 COMMAND PROCESSOR SUB-INTERFACE

This sub-interface will provide the following services to the service management command processors.

1. Determine if the IDS includes an entry for a given entity. The two part entity identification must be used to make this determination. One possible use of this will be to detect an attempt to define an entity which has already been defined.

2. Add or delete an entry in the IDS. This may be used when an entity or service is defined or cancelled.

3. Determine the status of an entry in the IDS.

4. Update the status of an entry in the IDS.

5. Load the module and execute an initialization procedure for an entity whose services are provided through the Service interface.

6. Register and de-register titles associated with an entity whose services are provided through the Service interface.

7. Association between command processors and the Service interface must be established via the XREF and XDCL mechanism. Command processors must obtain the services of the Service interface via procedure call/return method. The service interface must communicate with the command processors via intertask messages.

4.5.1.3 SERVICE USER SUB-INTERFACE

This sub-interface will provide the following services to the users of the services accessible through the Service interface.

1. Determine the status of an entity whose services are accessible through the Service Interface.

2. Establish an association with the Service Interface for obtaining the services provided by an entity whose services are accessible through the Service interface.

(e.g. provide a capability to open a SAP). Establishment of such an association must not require the associated entity to have been started.

3. Provide unsolicited indication about any change in the status of the entity for which an association has been established.

4. Delete the association established with the Service Interface.

5. Association between Service users and the Service interface must be established via the XREF and XDCL mechanism.

It should be noted that the Application and Gateway Service interface will not include the service user sub-interface.

4.5.1.4 SERVICE PROVIDER SUB-INTERFACE

This sub-interface will provide the following services to the entities which provide the services accessible through the Service interface.

1. Update status of the associated entry in the IDS.

2. Association between an entity which provides a service and the Service interface must be established via shared data structures.

4.5.2 DEFINE COMMAND

The following steps describe the function of define commands and the implementation guidelines for the associated command processors. Different things described in these steps should be done in the order in which they are listed.

If possible, the define command processors must use the services provided by the Service interface for the entity being defined, to provide these functions. These services are described in section 4.5.1.

1. Verify that the entity being defined has not already been defined. This should be done by checking if an entry exists in an Interface Data structure for the entity being defined.

4.0 RECOMMENDATION
4.5.2 DEFINE COMMAND

In some cases an IDS entry for a network solution, an interface or a management entity is automatically created if the media associated with it was used to load the system. In such cases, the duplicate definition is legal. The method to deal with this situation for an interface or a network solution has already been addressed in a separate DAP. A copy of that DAP is included as an appendix. Similar method should be used to deal with such duplicate definition of management entities.

2. If the definition of the entity is being specified in terms of another entity (for example, the definition of the X.25 gateway includes the name of one or more X.25 trunks), then verify that this other entity has already been defined. If this other entity has not been defined, then do not define the entity being defined currently. In other words, enforce the principal of bottom up definition.

3. If the entity which is to be defined, needs a physical resource, verify that the required physical resource is free and is in the correct state and status. Reserve the required physical resource for exclusive use by the entity being defined. Use the common routines "get_status_record" and "request_hardware_device" to do this. The state of a physical resource which may be reserved for exclusive use must be "on" or "down". The status of a physical resource which may be reserved for exclusive use must be "not configured".

4. The define command processor must create an IDS entry for the entity being defined.

5. The define command processors must support staring of an entity by calling a procedure provided by the associated start command processor. In other words, the start up code must not be duplicated.

6. If the entity being defined provides a management function (e.g. file access support), then the define command processor must also initialize and start the entity. The specific things which must be done as a part of starting of an entity are described in the next sub-section.

7. The define commands for entities which also have a corresponding start command must include an optional start parameter. This parameter will allow the command user to specify if the entity should be automatically started after it has been defined. The default value for this

parameter must always be true, i.e. start the entity.

4.5.3 START COMMAND

The following items describe the function of start commands and the implementation guidelines for the start command processors. Different things described in these steps should be done in the order in which they are listed.

If possible, the start command processors must use the services provided by the Service interface for the entity being started, to provide these functions. These services are described in section 4.5.1.

In general, all functions listed below should be provided by the start command processors. However, in some cases, it may be more appropriate for the entity being started to provide some of these functions. Such cases must be treated as exceptions and highlighted in appropriate design documents.

1. Make sure that the entity to be started has already been defined. Generate an error response if it has not been defined.

2. Make sure that the entity to be started has not already been started. Generate a warning response if this is the case.

3. Load the module for the entity being started if it needs to be loaded at the start up time (please see section 4.6 for load options).

4. If the entity to be started uses a physical resource, then the start command processor must verify that the state of the physical resource is "on" and its status is "configured". Also, if the entity is started successfully, then the start command processor must update the status of the physical resource in the appropriate hardware status table to be "active" or "enabled". This must be done by using the "get_status_record" and "put_status_record" common routines.

5. Execute the entity initialization procedure. This procedure may be provided by the command processor or included in the entity module. execution of this

procedure will start the entity module as a task, if necessary.

6. If the entity uses the services of other entities, establish the needed associations (i.e. open SAPs) with the Service interface(s) for these entities. Establishment of such an association does not require the entity or entities whose services are needed to have already been started (please also see item 2 of section 4.5.1.3).

   In other words an entity can be started even if the entities whose services it needs have not been started. However, in this case, the start command processor must include a warning in the command response. This warning should identify the need to start the entities whose services are needed. Also, in such cases the command response should state that the entity has been "enabled" instead of having been "started".

7. If the entity being started needs the services of entities in other systems, then the associations (e.g., establishment of a connection, initialization of a link like the HDLC link) needed to obtain such services must be established only if all needed services within the system are already available.

8. If the services provided by the entity are to be made known outside the system, register appropriate titles with the Directory. however, titles should be registered only after all needed services in the local system as well as other systems are available.

9. The start command processor must not start any other entity whose services may be required by the entity being started. However, if the entity being started is a group entity, then the start command processor must start all entities included in the group entity.

10. The start command must not be used to define an entity. In other words, the use of start command assumes that the entity being started has already been defined either explicitly or implicitly.

11. The start command must not include any parameter other than what is necessary to identify the entity being started.

12. The start command processors must be implemented in a way

so that they may be invoked from the define command processor for the same entity. This must be supported by providing a procedure call interface.

13. A single start command must be provided for entities of the same type. Valid types of CDCNET entities are listed in section 4.1.

4.5.4 CHANGE COMMAND

The following steps describe the function of change commands and the implementation guidelines for the associated command processors.

1. If a define command exists for the entity, then the change command must allow its users to change values of all parameters which can be specified via the define command except for the one used to identify the entity, i.e. its name.

2. It may not be possible to change the values of some of the parameters if an entity has already been started. Therefore, the change command processor must include code to determine which parameters can be changed when.

3. The change command processor must not change the value of any parameter if it can not change the value of all parameters for which a change has been requested via the change command. However, the command response must identify as many parameters as possible whose value can not be changed. In other words, the command processor must not quit after finding the first parameter whose value can not be changed.

4.5.5 STOP COMMAND

The following steps describe the function of stop commands as well the implementation guidelines for the associated command processors. Different things described in these steps should be done in the order in which they have been listed.

If possible, the stop command processors must use the services provided by the Service interface for the entity

being stopped, to provide these functions. These services are described in section 4.5.1.

In general, all functions listed below should be provided by the stop command processors. However, in some cases, it may be more appropriate for the entity being stopped to provide some of these functions. Such cases must be treated as exceptions and highlighted in appropriate design documents.

1. Delete any titles which were registered on behalf of the entity during its startup or after it had been started.

2. Inform the Service interface of the entity about the change in entity's status from "active" or "enabled" to "configured".

3. If the entity to be stopped was using the services of other entities, break the associations (e.g. close the SAPs) which were established to obtain these services.

4. If the entity uses a physical resource, update the status of the physical resource in the appropriate hardware status table to be "configured". This must be done by using the "put_status_record" common routine.

5. Execute the entity termination procedure. This procedure may be provided by the command processor or included in the entity module. This procedure must do general clean up by doing things like releasing memory and buffers assigned to the entity and canceling active timers started by the entity.

6. Identify the software module associated with the entity to be "deloadable".

7. A single stop command must be provided for entities of the same type. Valid types of CDCNET entities are listed in section 4.1.

4.5.6 CANCEL COMMAND

The following steps describe the implementation guidelines for the cancel command processors.

If possible, the cancel command processors must use the

services provided by the Service interface for the entity being cancelled. These services are described in section 4.5.1.

1. An entity must be stopped before it may be cancelled. Therefore, the cancel command processor must check the status of the entity (in the associated entry in IDS). If the status is "active" or "enabled", the command processor must generate a response stating that the entity must be stopped before it can be cancelled.

2. Delete the entry in the IDS for the entity. Also, release the memory being used for the data structure which contains the configuration information (value of configurable attributes) for the entity.

3. If the entity being cancelled had reserved a physical resource, the cancel command processor must release this resource by updating its status to be "not configured". Common routine "put_status_record" must be used to do so.

4. An entity must not be canceled if other entities exist whose definition includes a reference to or information about it, i.e., the entity being cancelled.

5. If an entity can not be canceled due to any one of the above reasons, it will be sufficient to give a generic response. For example, if the entity can not be canceled because its user entities have open SAPs, then there is no need for the command response to provide a list of entities who have open SAPs.

4.6 MISCELLANEOUS RECOMMENDATIONS

4.6.1 LOADING OF SOFTWARE TO SUPPORT ENTITIES

There are two options for the loading of the software which supports an entity. Under the first option, the software must be loaded as soon as the entity is started. Under the second option, the software is loaded when the service to be provided by the entity is truly needed. The second option requires presence of some software which can detect the need for the service provided by the entity. This software may be a part of the Service interface associated with the entity.

Both of these options are acceptable. Any one of these

options may be used based on the type of the service and its Service interface.

4.6.2 ORDER OF COMMAND EXECUTION

The following guidelines must be used to determine the order in which different entities which either use or provide services to each other must be defined, started, stopped and canceled.

1. The entities must be defined bottom up. In other words, the entity which will be referenced to in the definition of another entity must be defined before this other entity is defined.

2. The entities must be canceled top down. In other words, the entity which is defined in terms of another entity must be cancelled before this other entity is cancelled.

3. Different entities may be started or stopped in any order.

4. An entity must be stopped before it may be cancelled.

The implementation guidelines for different commands as described in previous sections will support these ordering guidelines.

4.6.3 GENERATION OF COMMAND RESPONSE

A command response should be generated only when all functions to be provided by the command processor has been completed. Only exceptions to this must be the functions which require services from other systems. Some examples of such functions are obtaining title translations, opening connections and establishing links. Even in these cases, an effort should be made to complete these functions before returning a response. However, since these functions may take an indeterminate amount of time to complete, a timer should be used to avoid long delay in returning a response. A thirty second timer is recommended.

4.7 IMPLEMENTATION RECOMMENDATIONS

This section provides recommendation for the implementation of the design direction included in this DAP.

1. The lower layer service interface provided by Generic 3A (Intranet) should be enhanced to provide the services described in section 4.1.

2. A separate design and development activity should be initiated to implement the Application and Gateway as well as the Interface and Layer entity Service interfaces.

3. All new implementation of command processors must be based on the guidelines provided in this DAP.

4. Update of existing commands to meet the guidelines being recommended in this DAP is not required. However, if a significant rework is being done in an existing command processor due to other reasons, then this rework must also include the changes needed to support the new guidelines. It will be responsibility of the design group to identify to management when a planned rework in an area should include the changes needed to support the new guidelines.

5.0 SUPPORTING EVALUATION

5.0 SUPPORTING EVALUATION

5.1 COMPATIBILITY IMPACT

The only compatibility implication of guidelines included in this DAP will come from the elimination of define and cancel commands for trunks. New guidelines will require a trunk to be included within a network solution or interface group entity. This should be addressed by doing the following:

- Retain the define and cancel commands for trunks for two to three releases after the commands for the associated group entity are released.

- Implement the define command processors for the network solutions and interfaces as new logical entities in a way which requires the trunk related information to be provided via optional parameters.

- Implement the define command processors for the network solution and interface group entities so that they will check if underlying trunk has already been defined using a define command for the trunk. If it has been, the command processor will use the configuration information in the LIB instead of expecting it to be provided via command parameters.

- Implement the cancel command processors for the network solution and interface so that they will cancel the definition of the underlying trunk also. Therefore, the use of old commands to cancel a trunk will only result in a warning response stating that the trunk has already been cancelled.

5.2 ALTERNATE SOLUTIONS/APPROACHES

The following is a brief description of different options which were considered for various issues addressed in the DAP.

ISSUE 1	SHOULD THERE BE A START PARAMETER ON THE DEFINE COMMAND TO OPTIONALLY START THE ENTITY BEING DEFINED
OPTION 1	ALLOW AN OPTIONAL START PARAMETER ON THOSE DEFINE COMMANDS WHERE IT MAKES SENSE. OBVIOUS BENEFITS OF THIS OPTION ARE: FEWER COMMANDS IN CONFIGURATION FILE EASE OF USE, I.E. ONE DOES NOT NEED TO TYPE TWO COMMANDS TO DEFINE AND START A NEW ENTITY FEWER COMMANDS IN THE CONFIGURATION FILE HELPS TO DELAY IMPLEMENTATION OF START COMMANDS
OPTION 2	DO NOT ALLOW THE START PARAMETER ON ANY DEFINE COMMAND. THE REASONS FOR THIS OPTION ARE: DOES NOT VIOLATE THE ORIGINAL OBJECTIVE THAT ONE COMMAND/ONE VERB MUST PROVIDE A SINGLE FUNCTION DOES NOT INTRODUCE INCONSISTENCY WHERE SOME DEFINE COMMANDS HAVE THIS PARAMETER AND SOME DO NOT ONE CAN ENHANCE START COMMANDS TO START MORE THAN ONE ENTITY AT THE SAME TIME HOW OFTEN AN OPERATOR IN A PRODUCTION ENVIRONMENT WILL TAKE ADVANTAGE OF THE START PARAMETER IS NOT OBVIOUS
RECOMMENDATION	OPTION 1 IS BEING RECOMMENDED IN THE DAP.

5.0 SUPPORTING EVALUATION
5.2 ALTERNATE SOLUTIONS/APPROACHES

ISSUE 2	SHOULD A STOP PARAMETER BE ALLOWED ON A CANCEL COMMAND
OPTION 1	YES DO ALLOW SUCH A PARAMETER. THE BENEFITS ARE: OPERATORS WILL HAVE TO TYPE IN ONE LESS COMMAND OPERATORS WILL NOT HAVE TO WORRY ABOUT WHAT MUST BE STOPPED
OPTION 2	DO NOT ALLOW SUCH A PARAMETER. REQUIRE THE OPERATOR TO EXPLICITLY STOP THE ENTITY BEFORE CANCELLING IT. THE REASONS FOR THIS ARE: DOES NOT VIOLATE THE ORIGINAL OBJECTIVE THAT ONE COMMAND/ONE VERB MUST PROVIDE A SINGLE FUNCTION HOW OFTEN AN OPERATOR IN A PRODUCTION ENVIRONMENT WILL TAKE ADVANTAGE OF THE STOP PARAMETER IN THE CANCEL COMMAND IS NOT OBVIOUS. FOR THAT MATTER, HOW OFTEN ONE WILL USE THE CANCEL COMMAND IN A PRODUCTION SYSTEM. DO WE WANT TO INTRODUCE UNNECESSARY COMPLEXITY (IN THE MEANING OF COMMANDS) FOR LOW FREQUENCY OPERATION.
RECOMMENDATION	OPTION 2 IS BEING RECOMMENDED IN THE DAP.
ISSUE 3	WHAT TYPE OF LINKAGE IS NEEDED BETWEEN A SERVICE USER AND A SERVICE PROVIDER. THINGS OF CONCERN ARE (1) ORDER IN WHICH ENTITIES MAY BE STARTED AND STOPPED (2) LOADING OF ENTITIES WHOSE SERVICES ARE NEEDED
OPTION 1	COMPLETELY DECOUPLE LOADING, STARTING AND STOPPING OF SERVICE USER AND SERVICE PROVIDER ENTITIES. PROVIDE AN INTERFACE LAYER OR PROCESS TO FACILITATE

DETERMINATION AND NOTIFICATION OF STATUS (AVAILABILITY/LACK OF AVAILABILITY) AS WELL AS ESTABLISHMENT OF ASSOCIATION NEEDED FOR THE USE OF THE SERVICE. THE FOLLOWING ARE SOME METHODS WHICH MAY BE USED 0 PROVIDE SUCH AN INTERFACE LAYER OR PROCESS. THE FOLLOWING ARE SOME METHODS WHICH MAY BE USED TO SUPPORT THIS OPTION.

1. AN ACTUAL INTERFACE LAYER LIKE 3A

2. EDB TYPE DATA STRUCTURE AND ASSOCIATED COMMON ROUTINES TO (a) DETERMINE STATUS OF THE SERVICE PROVIDE, (b) REGISTER INTENT TO USE THE SERVICE BY PROVIDING ADDRESS OF AN STATUS INDICATION PROCEDURE, (c) NOTIFICATION WHEN THE SERVICE BECOMES AVAILABLE ALONG WITH ADDRESS OF THE SAP MANAGEMENT PROCEDURE.

3. PASSIVE TITLE TRANSLATION TO LEARN WHEN SERVICE BECOMES AVAILABLE AND TO OBTAIN THE ADDRESS OF THE SAP MANAGEMENT PROCEDURE.

OPTION 2

COUPLE LOADING, STARTING AND STOPPING OF SERVICE USER AND SERVICE PROVIDER ENTITIES. THE FOLLOWING ARE SOME METHODS WHICH MAY BE USED TO SUPPORT THIS OPTION.

1. ENFORCE AN ORDERING REQUIREMENT FOR STARTING ENTITIES. THE SERVICE PROVIDER MUST BE STARTED BEFORE THE SERVICE USER. IN THIS CASE THE ADDRESS OF THE SAP MANAGEMENT PROCEDURE OF THE SERVICE PROVIDER MAY BE (a) LINKED IN VIA XREF, (b) OBTAINED VIA TITLE TRANSLATION OR (3) OBTAINED FROM AN IDS ENTRY

2. DO NOT ENFORCE ORDERING REQUIREMENT. HOWEVER WHEN A SERVICE USER IS STARTED, ALSO AUTOMATICALLY START THE SERVICE PROVIDER.

3. LINK THE SAP MANAGEMENT ROUTINE IN SERVICE PROVIDER MODULE TO APPROPRIATE ROUTINE IN THE SERVICE USER. THIS WILL FORCE THE SAP MANAGEMENT PROCEDURE OF SERVICE PROVIDER (AND POSSIBLY MOST OF

	THE SERVICE PROVIDER MODULE) TO BE LOADED WHEN THE SERVICE USER MODULE IS LOADED.
RECOMMENDATION	USE OPTION 1 WITH METHOD 1.
ISSUE 4	HOW SHOULD AN ENTITY OBTAIN THE ADDRESS OF THE SAP MANAGEMENT PROCEDURE OF AN ENTITY WHOSE SERVICES ARE REQUIRED.
OPTION 1	USE THE TITLE TRANSLATION SERVICE OF DIRECTORY AS DISCUSSED BEFORE.
OPTION 2	HAVE THE SAP MANAGEMENT PROCEDURE LINKED IN. IF AN OPTION WAS TO BE MADE AVAILABLE TO AN ENTITY TO USE A CONFIGURABLE SERVICE (E.G. XNS TRANSPORT OR ISO TRANSPORT), THEN OPTION 2 WILL REQUIRE LOADING OF POTENTIALLY UNNECESSARY SOFTWARE.
OPTION 3	USE THE SERVICES PROVIDED BY A SERVICE INTERFACE LAYER. THE ADDRESS OF THE SERVICE INTERFACE LAYER PROCEDURES SHOULD BE KNOWN VIA XREF.
RECOMMENDATION	USE OPTION 3.
ISSUE 5	WHEN SHOULD THE SOFTWARE NEEDED TO PROVIDE THE SERVICES SUPPORTED BY AN ENTITY MUST BE LOADED
OPTION 1	LOAD IT WHEN IT IS TO BE USED. THE BENEFITS ARE: BETTER MEMORY UTILIZATION SUPPORTS OVER COMMITTMENT OF MEMORY
OPTION 2	LOAD IT WHEN THE ENTITY IS STARTED. THE BENEFITS ARE: DOES NOT REQUIRE THE SOFTWARE WHICH IMPLEMENTS THE ENTITY TO BE DIVIDED INTO TWO PARTS. ONE PART IS USED TO DETECT THE NEED, THE OTHER PART IS USED TO PROVIDE THE SERVICE.

5.0 SUPPORTING EVALUATION
5.2 ALTERNATE SOLUTIONS/APPROACHES

CAN RESULT IN A DELAY IN PROVIDING THE SERVICE DUE TO SLOW FILE ACCESS.

MAY NOT BE ABLE TO PROVIDE THE SERVICE IF THE NEEDED SOFTWARE CAN NOT BE LOADED.

OPTION 3 — USE OPTION 1 OR 2 AS APPROPRIATE.

RECOMMENDATION — USE OPTION 3.

ISSUE 6 — SHOULD THE DEFINE COMMAND BE ALLOWED TO COMPLETE SUCCESSFULLY, IF STATE OF THE PHYSICAL RESOURCE NEEDED BY THE ENTITY IS "OFF"

OPTION 1 — YES, DO ALLOW THE DEFINE COMMAND TO COMPLETE SUCCESSFULLY. THE BENEFITS ARE:

WILL ALLOW TO CHANGE THE STATE OF THE ELEMENT AT A LATTER TIME AND ALLOW ONE TO USE IT WITHOUT HAVING HAVE TO ENTER THE DEFINE COMMAND AGAIN.

OPTION 2 — NO, DO NOT ALLOW THE DEFINE COMMAND TO COMPLETE SUCCESSFULLY. THE BENEFITS ARE:

STATE OF A PHYSICAL RESOURCE IS "OFF" BECAUSE IT IS EITHER PHYSICALLY NOT PRESENT OR THE OPERATOR WANTS IT TO BE TREATED IT THAT WAY. WHY ALLOW THE DEFINITION OF AN ENTITY WHICH IS TO USE A RESOURCE WHICH IS PHYSICALLY NOT PRESENT.

IF AN OPERATOR WANTS TO TAKE IT OUT OF SERVICE TEMPORARILY, HE CAN CHANGE THE STATE TO BE "DOWN". THIS WILL ALLOW THE DEFINITION OF A POTENTIAL USER TO COMPLETE SUCCESSFULLY.

HELPS US NOT END UP SUPPORTING UNNATURAL SITUATIONS, SIMPLY BECAUSE WE THINK IT WILL BE NICE TO DO SO.

RECOMMENDATION — USE OPTION 2.

6.0 REQUIRED ACTIONS

6.0 REQUIRED ACTIONS

ACSD management should schedule the implementation of the design direction provided in this DAP according to the implementation recommendation described in section 4.6.

APPENDIX A

COPY OF RE-CONFIGURATION DAP

GUIDELINES FOR THE DEFINE/START/CHANGE/STOP/CANCEL COMMANDS

RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTE

AND THE INTERFACE BETWWEN A COMMAND PROCESSOR AND SSR

B. S. Sekhon
01/11/85

DISCLAIMER: This document is an internal working paper only. It is unapproved and subject to change, and does not necessarily represent any official intent on the part of Control Data Corporation.

GUIDELINES FOR THE DEFINE/START/CHANGE/STOP/CANCEL COMMANDS

REVISION RECORD			
Revision	Description	Author	Date
A	Original version,	BSS	11/13/84
B	Update to resolve the TDRB comments	BSS	01/11/85

A1.0 INTRODUCTION

A1.0 INTRODUCTION

This DAP addresses two problems. The first problem is concerned with the configuration (via commands in the configuration file or on line commands) of a trunk or an interface which has already been configured by the minimum task set command processor (also known as the boot command processor). The second problem is concerned with the nature and timing of the response provided by the command processors used to configure a trunk or an interface as well as with the method used by an SSR to inform the command processor about the result of the configuration request.

A2.0 SITUATION REQUIRING ACTION

A2.0 SITUATION REQUIRING ACTION

A2.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

When a system gets loaded, the minimum task set command processor configures the trunk or interface (in the case of the C170 MCI) which was used to load the system. The minimum task set command processor uses default values for all configurable parameters. Next, the configured trunk or interface is used to obtain the configuration file for the system. In general this file will include a command to configure a trunk or interface using the media which was used to load the system.

Normally a command which tries to configure an already configured trunk or interface, is given a failure status indicating the fact that the trunk or interface is already configured. However, in the situation described above, the configuration command should be accepted and the trunk or interface should be re-configured.

A2.2 CONFIGURATION COMMAND RESPONSE SUPPORT

At present the command processors used to start or stop a trunk do not wait for the completion of the start up or stop process. A command completion response is generated as soon as the start up or stop inter task message is sent to the SSR. The command processors should wait till the SSR has successfully started or stopped the trunk or has timed out trying to do so, and then generate a more meaningful response.

Also, at present the method to be used by an SSR to inform the command processors about the status of their requests has not been defined.

A3.0 CONSTRAINTS AND REQUIREMENTS

A3.0 CONSTRAINTS AND REQUIREMENTS

The following are the requirements and constraints which must be met by any solution to address the situation described in the previous section.

- The solution should make sure that no input or output data already queued up for the trunk is lost due to its re-configuration.
- The solution should not impose any new requirements on the order in which the configuration commands are to be processed.
- The solution should not require the command processor to run in the no pre-emption mode.
- The command processor used to start up or stop a trunk should protect itself from the situation when the SSR fails to generate a response due to an unexpected error.

A4.0 RECOMMENDED TECHNICAL SOLUTION

A4.0 RECOMMENDED TECHNICAL SOLUTION

A4.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

The following is a step by step description of the recommended solution to re-configure the trunk or interface which was used to load the system and subsequently configured by the minimum task set command processor.

- Two new flags should be added to the HDLC, MCI and ESCI LIBs. One of these flags (called flag1 in the rest of this document) should be used to indicate the fact that the LIB was created by a minimum task set command processor and is associated with the media which was used to load the system. The second flag (called flag2 in the rest of this document) should be used to indicate the fact that another LIB exists for the media associated with this LIB (i.e. the one in which flag2 is set).

- The minimum task set command processors should be changed to set the flag called flag1 in the LIB created by it to configure the trunk or interface which was used to load the system.

- The command processors used to define a trunk or an interface should be changed to check if the media to be used for the trunk or interface is already in use. If the answer is yes, then the command processor should locate the associated LIB and check if that LIB was created by a minimum task set command processor. If the answer to this check is also yes, the command processor should further check if any non-default (except the name of the trunk or interface) values have been specified for the configurable options. If no non-default values have been specified, the command processor should return a completion status along with an informational message saying that the trunk or interface was defined and started by the minimum task set command processor. On the other hand, if non-default values have been specified, the command processor should create a new LIB and set the flag called flag2 (defined earlier) in it.

- The command processor or code used to start a trunk or an interface should check if the flag called flag2 is set in the LIB for the trunk or interface. If this flag is not set, then the normal start up process should be followed.

A4.0 RECOMMENDED TECHNICAL SOLUTION
A4.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

- On the other hand if flag2 is set in the LIB, the command processor should check if another LIB exists for the trunk to be started. If no other LIB exists, the normal start up process should be followed. Otherwise, the command processor should send a stop function to the SSR which is currently driving the trunk and wait for its completion. After receiving an indication about the completion of the "stop" function, the command processor should move the pointer to the output queue and related information (about the output queue) from the old LIB to the new LIB; and send the "start up" function to the SSR associated with the old LIB. Next the command processor should delete the old LIB and any associated NIB.

- If the command processor or code used to start a trunk or an interface detects that it will have to send a "stop" function and follow it up with a "start up" function, then it should increase the priority of its task to the highest possible value. The reason for this is to minimize the time during which the trunk or interface will be unavailable for data transfer.

A4.2 CONFIGURATION COMMAND RESPONSE SUPPORT

The following is a step by step description of the process which will allow a command processor to learn the status of its request to an SSR.

- Each command processor which sends a function to an SSR should include its task id along with the function code in the inter-task message sent to the SSR.

- The SSR should provide a response to the command processor via an inter-task message. This response should consist of a status code.

- The command processor should use a time out timer to protect itself from the situation where the SSR fails to provide a response due to a software bug, etc. The time out value should take into account the time required by the SSR to complete the requested function.

A5.0 SUPPORTING EVALUATION

A5.0 SUPPORTING EVALUATION

A5.1 ALTERNATE SOLUTIONS/APPROACHES

A5.1.1 RE-CONFIGURATION OF THE TRUNK/INTERFACE USED TO LOAD THE SYSTEM

One alternative considered in this area was to not to support the re-configuration of a trunk which was configured by the minimum task set command processor. Since, in general, one can not control the media over which a given DI may get loaded, this alternative will prohibit configuration of a trunk or an interface with non-default values for its configurable options. If this is the case, then why should we have any configurable options at all. Because of this reason, this alternative is not being recommended.

Another alternative considered was to define a new SSR function called re-configuration. This function would have simulated a "stop" function followed by a "start" function. This alternative would have required more changes in the SSRs than the ones required by the recommended solution. Therefore this alternative is not being recommended.

A5.1.2 CONFIGURATION COMMAND RESPONSE SUPPORT

The only alternative considered in this area was to have the command processor periodically examine the status of the trunk or interface in the LIB, and use it to determine the type of response to be generated. This alternative is not being recommended because the trunk or interface status in the LIB does not and should not include all reasons why a trunk or interface can not be started or stopped, etc.

A6.0 REQUIRED ACTIONS

A6.0 REQUIRED ACTIONS

This DAP closes the subject design issue. The ACSD development should implement the recommended solution. It should be noted that the code needed to support most of the items in the recommended solution, already exists.