

Defining Terminals and Printers

This chapter describes Com-plete's Terminal Information Block Table (TIBTAB).

In the vast majority of environments it is preferable to have Com-plete build the TIBTAB dynamically during startup. All required information about terminals and printers connected using the VTAM access method can be obtained by Com-plete directly from VTAM during LOGON / SIMLOGON.

TCP/IP based connections (Telnet tn3270 sessions and connections to LPD print servers) are supported using dynamically allocated TIBs only, not for TIBs hard-coded in a TIBTAB.

The information in this chapter applies to those installations only which still must maintain and assemble a coded TIBTAB for whatever reason.

This chapter covers the following topics:

- Overview
 - How to Code TIBTAB
 - How to Create TIBTAB
 - Printout Spooling TIBTAB Considerations
 - Batch Output Facility (SYSOUT)
 - Logical Output Drivers
 - Line Printer Daemon (LPD) Protocol
-

Overview

The Terminal Information Block Table (TIBTAB) defines the terminals and lines to be used by Com-plete. TIBTAB resides as a load module in the load library, and is loaded by the Com-plete control program during initialization.

Since the TIBTAB exists as a member of the load library/core image, the maximum number of characters that can be used to make up the name is eight. Multiple copies under separate names can exist.

When you create a new TIBTAB, it is recommended that the module name used be different from the module name of the current working TIBTAB. If the two module names are the same and an undetected error exists in the newly created TIBTAB, Com-plete initialization will fail.

How to Code TIBTAB

The TIBTAB is coded as an Assembler language module using the following macros and compiler instructions:

- TIBSTART
- TIB
- Translation Table Definitions
- CMDEVS
- TIBEND
- END

These instructions are combined to create a non-executable module, which is nothing more than a table.

A sample TIBTAB is supplied in the source library (member TIBTAB).

TIBSTART Macro

The TIBSTART macro defines the number of Terminal Information Block (TIB) entries to be generated.

A TIB entry is required for each terminal in the network. In addition, a TIB entry is required for each simultaneously-executing batch job that uses a Com-plete function, as well as for each simultaneously-executing online program invoked by the ATTACH function of Com-plete.

The format for using the TIBSTART macro is:

```
name TIBSTART NOTIBS=n
  [ ,SMC=(c1,c2,...,cn) ]
  [ ,RMC=(c1,c2,...,cn) ]
  [ ,VAL={YES|NO} ]
```

The arguments are:

name

Required.

Specifies the TIBTAB name.

NOTIBS=n

Required.

Specifies the number of TIB entries to be generated. This number also specifies the largest number allowed when specifying a TID number (TIB macro).

Note that the value specified for NOTIBS must be at least as large as the total number of TIB macros defined in TIBTAB and all the dynamically-allocated ACCESS terminals in the network. If no batch TIBS are specifically defined, then NOTIBS should also be large enough to allow one or more undefined TIB entries to exist.

SMC=(c1,c2,...,cn)

Optional.

Default	SMC=(1)
---------	---------

Specifies the default authorization-sending message class codes to be assigned to all defined printer TIBs without an SMC argument assigned. Authorization message class codes are fully described in the Com-plete Utilities documentation.

RMC=(c1,c2,...,cn)

Optional.

Default	RMC=(1,2,4)
---------	-------------

Specifies the default authorization receiving message class codes to be assigned to all defined printer TIBs without an RMC argument assigned. Authorization message class codes are fully described in the Com-plete Utilities documentation.

VAL=[YES|NO]

Optional.

Default	VAL=YES
---------	---------

Specifies whether or not the TIB parameters are validated. Note that assembly of the TIBTAB is faster if validation of the parameters is bypassed. Care must be taken, however, to ensure that the parameters are correct.

VAL=YES indicates that the parameters are validated.

VAL=NO indicates that the parameters are not validated.

TIB Macro

The TIB macro defines a terminal and its characteristics.

Batch jobs and attached online programs that use Com-plete functions also require use of a TIB entry. TIB entries for batch jobs and attached online programs can be defined explicitly by use of the batch device specification defined below, or are defaulted by specifying a NOTIBS value that is larger than the total number of TIB and LGCB macros defined.

The format for using the TIB macro is:

```
[name] TIB tid, {VTAM|ACCESS}, device
[ ,ALTTID=tid]
[ ,CTLTIB={YES|NO}]
[ ,DEL={YES|NO}]
[ ,FORMAT=n]
[ ,GROUP=ALL=NO]
[ ,HC={YES|NO}]
[ ,LEN=n]
[ ,LINES=n]
[ ,LOGOFF={YES|NO}]
```

```
[ ,LOWER={YES|NO}]
[ ,MAXRU=]
[ ,NAME=name]
[ ,NODEID=node]
[ ,OPT= ( {ALTE|NOALTE}
, {PAD|NOPAD}
, {MOD3|NOMOD3}
, {EBCDIC|BCDIC|EBCDIC,ADD|BCDIC,ADD}
, {CR|NOCR}
, {EM3270|NOEM3270}
, {SHARE|NOSHARE}
, {ACQUIRE|NOACQUIRE}
, {COMP|NOCOMP}
{SCS|NOSCS}
{IBM|FACOM|HITACHI}) ]
[ ,PRI=x0|1|2|3|z]
[ ,RMC=(c1,c2,...,cn)]
[ ,SCHC=tid]
[ ,SMC=(c1,c2,...,cn)]
[ ,STALL={YES|NO}]
[ ,TRTABS=trantab]
[ ,VAL={YES|NO}]
```

The arguments are:

name

Optional.

Specifies the TIB name; for assembly listing purposes only.

The name can be any character string that is valid as an Assembler label.

tid

Required.

Specifies the Terminal Identification number of the terminal being defined. This value must be an integer from 1 to 9999, and must be less than the value for NOTIBS as specified in the TIBSTART macro.

VTAM or ACCESS

Required.

Specifies the access method used for the device.

device

Required.

Specifies the device type of the terminal being defined. This value must be one of the values defined in the terminal device type table in Appendix.

If the NOTIBS argument of the TIBSTART macro specifies a value larger than the total number of TIB, the excess TIBs are reserved for programs invoked by the ATTACH function, and VTAM terminals.

ALTTID=tid

Optional.

Default	None
---------	------

Specifies the TID number of the terminal to receive messages when the terminal being defined is inoperative, or when more than 10 messages are queued for the terminal being defined.

CTLTIB={YES|NO}

Optional.

Default	CTLTIB=NO
---------	-----------

Specifies whether or not the terminal being defined is to be assigned control status. Control terminals have privileged status and are allowed to execute Com-plete utility programs and certain functions reserved for the control user.

Note that at least one terminal must be defined as having control status. This should be the terminal with a TID value of one.

CTLTIB=YES indicates that the terminal being defined has control status.

CTLTIB=NO indicates that the terminal being defined has non-control status.

DEL={YES|NO}

Optional.

Default	DEL=NO
---------	--------

Specifies whether the terminal being defined is to be brought up deleted when Com-plete is initialized.

DEL=NO specifies that the terminal is to be available for use when Com-plete is initialized.

Since Com-plete allows any VTAM terminal to log on regardless of whether it is in the TIBTAB or not, in order to disallow logon from a VTAM terminal, that terminal must be included in the TIBTAB with DEL=YES specified.

FORMAT=n

Optional.

Default	Depends upon the device type as specified in Terminal Device Type Codes
---------	---

Specifies the format to be used for CRT devices.

Here, *n* must be specified as one of the following:

- 240(6x40)
- 480(12x40)
- 960(12x80)
- 1920(24x80)
- 2560(32x80)
- 3440(43x80)
- 3564(27x132)

The arguments **LINES** and **LEN** are automatically set by the value given for the **FORMAT** argument as described by the numbers in the parentheses.

Note that if the **FORMAT** argument is specified in a TIB, **LINES** and **LEN** cannot be specified.

GROUP=ALL=NO

Optional.

Default	The associated TID will reside in the ALL grouping.
---------	---

ALL=NO indicates that this TID will not be considered part of the ALL grouping by message switching. This includes **CALL MESGSW**, the hello message, and ***UM**. The ALL grouping can also be used in operator commands.

HC={YES|NO}

Optional.

Default:

HC=YES for all devices except 3270-type terminals; **HC=NO** for 3270-type terminals.

Specifies whether or not the terminal being defined is a hardcopy device.

HC=YES indicates that the terminal is a hard copy device. **NO** indicates that the terminal is not a hardcopy device.

LEN=n

Optional.

Default	The minimum and maximum values will depend upon the device type.
---------	--

Specifies the length of the line to be used.

Note:

Do not specify if FORMAT is specified.
Here, n must specify a decimal value.

LINES=n

Optional.

Default	The minimum and maximum values will depend upon the device type.
---------	--

Specifies the maximum number of lines to be used in a display or output.

Note:

Do not specify if FORMAT is specified.
Here, n must specify a decimal value.

LOGOFF={YES|NO}

Optional.

Default	LOGOFF=YES
---------	------------

Specifies whether or not this terminal will be logged off after the period of inactivity defined by the AUTOLOGOFF sysparm.

LOGOFF=YES indicates that the terminal will be logged off for inactivity.

LOGOFF=NO indicates that the terminal will not be logged off for inactivity.

LOWER={YES|NO}

Optional.

Default	LOWER=NO
---------	----------

Specifies whether or not lower-case characters will be accepted from the device as input.

LOWER=YES indicates that all characters will be accepted in lower-case format.

LOWER=NO indicates that only upper-case characters are supported. For input, lower-case characters are translated to upper-case.

MAXRU=n

Optional.

Default	None
---------	------

Specifies an override RUSIZE for a VTAM device.

NAME=name

Optional.

Default	CL8'TIB0nnnn' where nnnn is the four-digit TID number padded with leading zeros if necessary.
---------	---

Specifies a one- to eight-character name to be associated with this TIB. If VTAM is the access method being used, the name must specify the name of the VTAM node (LU) to which Com-plete is to connect (see the ACQ keyword).

Here, *name* is also used in group name searches. TIB name entries will be searched before searching for a group name. The first TIB name will satisfy the search, and the TID of the TIB with that name will be used. If duplicate TIB names are defined, only the first will be found. If a duplicate TIB name and group name are defined, only the first TIB name will be found.

NODEID=node

Optional.

Default	None
---------	------

Specifies the ACCESS node number. This value must be an integer between 1 and 65536. Valid only if ACCESS is specified for the CUU parameter.

Specifies the optional features to be assigned to a terminal.

OPT=value

Format (default is underlined>):

OPT= ({MOD3|NOMOD3}, {PAD|NOPAD}, {ALTE|NOALTE},
 {EBCDIC|BCDIC|EBCDIC,ADD|BCDIC,ADD}, {CR|NOCR},
 {EM3270,NOEM3270}, {SHARE|NOSHARE}, {ACQUIRE|NOACQUIRE},
 {COMP|NOCOMP}, {SCS|NOSCS}, {IBM|FACOM|HITACHI})

Meaning of the values:

MOD3|NOMOD3

Optional.

Default	<u>OPT=NOMOD3</u>
---------	-------------------

Specifies whether or not the IBM 3275 terminal being defined has an IBM 3284 model 3 printer attached.

OPT=MOD3 indicates that the IBM 3275 terminal has an IBM 3284 model 3 printer attached.

OPT=NOMOD3 indicates that the IBM 3275 terminal does not have an IBM 3284 model 3 printer attached.

PAD|NOPAD

Optional.

Note:

This operand is ignored if the terminal being defined is not a TTY device.

Default	OPT=NOPAD
---------	-----------

Specifies whether padding (idle) characters should be added to the line protocol output in order to allow for mechanical motion (e.g., carriage return). Because most terminals have internal buffering to account for mechanical motion so that padding is not required, OPT=NOPAD is normally specified.

OPT=PAD indicates that idle characters should be added to the line protocol.

OPT=NOPAD indicates that no idle characters will be added.

ALTE|NOALTE

Optional.

Default	If LINES=24 and LEN=80 or LINES=12 and LEN=40 are specified, NOALTE; for any other combination of LINES and LEN, ALTE; for 328x printers, OPT=NOALTE.
---------	---

Is applicable to terminals that support alternate screen sizes.

Specifies whether the alternate or standard erase write is used. Use of the alternate erase write generally results in a larger device buffer size.

OPT=ALTE indicates that the alternate erase write is used.

OPT=NOALTE indicates that the standard erase write is used.

EBCDIC|BCDIC|EBCDIC,ADD|BCDIC,ADD

Optional.

Note:

This option applies only to IBM 2740 model 1 terminals, and will be ignored if another terminal type is specified.

Default	OPT=EBCDIC
---------	------------

Specifies the type of keyboard being used.

OPT=EBCDIC indicates an EBCDIC keyboard.

OPT=BCDIC indicates a BCDIC keyboard.

OPT=EBCDIC,ADD indicates an EBCDIC keyboard with an adding machine.

OPT=BCDIC,ADD indicates a BCDIC keyboard with an adding machine.

CR|NOCR

Optional.

Default	OPT=NOCR
---------	----------

Specifies whether or not the carriage return is supported by the device being defined.

OPT=CR indicates that the carriage return is a supported feature.

OPT=NOCR indicates that the carriage return is not a supported feature.

EM3270|NOEM3270

Optional.

(Applies to TTY TIDs only.)

Default	OPT=NOEM3270
---------	--------------

Specifies that the TID should use 3270 emulation for a 3101 device. Refer to 3101 Terminal Support for further details.

SHARE|NOSHARE

Optional.

(Applies to VTAM printers only.)

Default	OPT=NOSHARE
---------	-------------

Specifies whether or not a printer can be shared among local copy functions, Com-plete, and other VTAM applications.

OPT=SHARE indicates a VTAM printer-shared mode. When a printer is identified as shared, Com-plete issues a CLSDST for the printer when there is no more output from the application program. The printer is then free to handle any local copy requests that may have been queued while the device was bound to Com-plete. When output is again available for the printer, Com-plete will reacquire the device via a SIMLOGON, print the data, and again issue a CLSDST for the device.

Note:

OPT=ACQUIRE must also be specified if OPT=SHARE is used.

OPT=NOSHARE indicates that a VTAM printer is not available for local copy requests.

ACQUIRE|NOACQUIRE

Optional.

(Applies to VTAM terminals only.)

Default	OPT=ACQUIRE
---------	-------------

Specifies whether or not the VTAM terminal is to be acquired when Com-plete VTAM startup options are initialized or when VTAM support is started with the VTAM START OC command.

OPT=ACQUIRE indicates that the terminal is acquired when Com-plete VTAM support is started (using the VTAM macro SIMLOGON). For additional information on required sysparm operands, see the VTAM startup options in the chapter entitled *Initialization - Com-plete Startup Procedure*.

OPT=NOACQUIRE indicates that the terminal is not acquired when Com-plete VTAM support is started.

COMP|NOCOMP

Optional.

Default	OPT=COMP
---------	----------

Specifies whether or not data streams being sent to 3270-type terminals (incl. printers) are to be compressed by replacing repeated characters with a repeat-to-address order, thereby reducing the amount of data transferred to the terminal.

Most 3270 and compatible terminals support this compression.

OPT=COMP indicates that compression is to be performed.

OPT=NOCOMP indicates that compression is not to be performed.

SCS|NOSCS

Optional.

Default	OPT=NOSCS
---------	-----------

Specifies whether or not the printer is an SCS device. OPT=SCS specifies an SCS printer. OPT=NOSCS specifies a normal printer.

Note that this option is only valid for ACCESS printers. Printers acquired via Com-plete VTAM will have this option overwritten by the information obtained from the bind during acquisition of the device.

IBM|FACOM|HITACHI

Optional

Default	OPT=IBM
---------	---------

In general, all terminals use the same 3270 command codes. However, some 3270 terminals use a proprietary set of codes. If a device you are defining uses Fujitsu terminal command codes, specify FACOM here, while for terminals using Hitachi command codes specify HITACHI.

PRI={0|1|2|3}

Optional.

Default	PRI=1
---------	-------

Specifies the dispatching priority to be assigned to the terminal being defined. The priority specified can be 0, 1, 2, or 3, where 0 is the lowest priority.

Attached terminals are automatically assigned a priority of 0, forcing them to have lowest priority.

RMC=(c1,c2,...,cn)

Default	The RMC value, if any, assigned in the TIBSTART macro; otherwise, RMC=(1,2,4).
---------	--

Specifies the authorization receiving message class codes to be assigned to this terminal.

Note that if a terminal user logs on to Com-plete, the RMC value assigned to the user ID will override the value specified in the TIB macro.

SCHC=tid

Optional.

(Valid for 3270-type terminals only.)

Default	None
---------	------

Specifies the TID number of the terminal to be designated as the hardcopy device for the terminal being defined.

This keyword operand provides support for the screen-to-hardcopy function of Com-plete, which is available for 3270-type terminals. Details on the use of this function can be found in the Com-plete Utilities documentation.

SMC=(c1,c2,...,cn)

Optional.

Default	The SMC value, if any, assigned in the TIBSTART macro; otherwise, SMC=(1).
---------	--

Specifies the authorization sending message class codes to be assigned to the terminal being defined.

Note that if a terminal user logs on to Com-plete, the SMC value assigned to the user ID will override the value specified in the TIB macro.

STALL={YES|NO}

Optional.

Default	STALL=NO
---------	----------

Specifies whether or not the TIB is stalled when Com-plete is started. No input is accepted from a stalled terminal, but Com-plete does poll the terminal and handles attention interrupts.

STALL=YES indicates that the terminal is stalled when Com-plete is initialized.

STALL=NO indicates that the terminal is unstalled when Com-plete is initialized.

The terminal can be stalled using the STALL operator command or unstalled using the UNSTALL operator command.

TRTABS=trantab

Optional.

Default	None.
---------	-------

Specifies the name of a load module containing the TIB depending translate tables. See copybook CCTR3270 for an example. This copybook maps the system-wide translate tables, which can be modified for general translation, which means CCTR3270 is the default translate table for all TIDs that have no TRTABS specified.

VAL={YES|NO}

Optional.

Default	The VAL coded on the TIBSTART macro, if specified; otherwise, VAL=YES.
---------	--

Specifies whether or not the TIB macro parameters are validated.

Note that assembly of the TIBTAB is faster if validation of the parameters is bypassed. Care must be taken, however, to ensure that the parameters are correct.

VAL=YES indicates that the parameters are validated.

VAL=NO indicates that the parameters are not validated.

CMDEVS Macro

The CMDEVS macro is an optional macro used internally by Com-plete and by the user to force an entry to be created in the device type table for a certain type of device. When a device type code is used in a TIB macro, a device type table entry is created automatically by the TIBEND macro.

In a VTAM environment, a TIB can be created for a VTAM LU that logs on to Com-plete. For the device to be operational, a device table entry must exist for that device. If no TIBs are defined with that device type, no entry exists in the device type table. The CMDEVS macro allows the user to force the entry to be created.

Note:

The CMDEVS macro must be defined before the TIBEND macro.

The format for using the CMDEVS macro is:

```
[name] CMDEVS GEN,TYPE=xxx
```

The arguments are:

name

Optional.

Specifies the CMDEVS statement name; for assembly listing purposes only.

Note that name can be any character string that is a valid Assembler tag symbol.

GEN

Required.

Indicates that the device type specified by the TYPE argument is to be added to the generation/definition of the TIBTAB.

TYPE=xxx

Required.

Specifies the type of device to be included in the TIBEND generation.

Here, *xxx* must specify a device type code as listed in Terminal Device Type Codes.

Note:

The TIBSTART macro automatically defines the following device types:
3270L, 3278L, 3284L, 3286L, 3287L, 3288L, TTY, BATCH, LU62S, LU62C, FSP.
This means that the CMDEVS macro need not be specified.

TIBEND Macro

The TIBEND macro defines the end of the TIBTAB.

Note:

TIBEND must immediately follow all TIB macro statements and (if any exist) all CMDEVS macro statements.

The TIBEND macro has several functions. Its primary function is to generate a device table entry for each device type that has been defined in a TIB macro. (Additional device types can be defined using the CMDEVS macro.)

There are no operands are specified for the TIBEND macro.

The format for using the TIBEND macro is:

```
TIBEND
```

END Statement

The END statement is an Assembler instruction that identifies the end of the source module definition.

Note:

The END statement must immediately follow the TIBEND macro statement.

The format for using the END statement is:

```
END
```

How to Create TIBTAB

After coding the desired TIBTAB, the table must be assembled and link edited into a Complete STEPLIB. It is recommended that the TIBTAB be page-aligned for the purpose of clarity during problem determination. To achieve this, use the PAGE linkage editor option.

The new TIBTAB will be available for use the next time Complete is initialized.

Note:

The TIBTAB being created must be expressly requested at initialization time by use of the sysparm TIBTAB.

Dynamic Completion of TIBTAB During Com-plete Initialization

VTAM and ACCESS terminals need not necessarily be defined in the TIBTAB load module. Instead, you can maintain a table of TIB definitions which is used by Com-plete during initialization to dynamically create or complete the TIBTAB. In this table, you can define TIBs with unique names, with or without fixed TIB numbers (TIDs). You can specify all parameters described for the TIB macro in this chapter. For details about the appropriate maintenance utility, refer to the description of UUTIL, function TT in the Com-plete Utilities documentation.

During initialization, depending on sysparm TIBTAB, either the TIBTAB load module is loaded or an empty TIBTAB is created. Afterwards, the TIBTAB is completed dynamically using this TIB definition table. At the first step, all TIBs defined with fixed TIDs are built, if the appropriate TIDs are still free. Override takes place only if TIB name and TID match, otherwise the dynamic TIB definition is skipped. At the second step, TIBs defined without a fixed TID are allocated using free TIBTAB entries. If a TIB with the same name already exists, it is overridden by the dynamic definition.

Printout Spooling TIBTAB Considerations

This section discusses printout spooling TIBTAB considerations.

Defining Virtual Printers

Use of the printout spooling facilities may require that TIB entries be inserted in the Com-plete TIBTAB for virtual printers. The following example shows the recommended structure of such entries:

```
TIB 131,VTAM,3288L,NAME=VIRTPRI1,DEL=YES,GROUP=(ALL=NO)
```

where:

VTAM is used to define a dummy TIB.

VIRTPRI1 is a sample printer name that can be used to access this virtual printer.

DEL=YES must be coded for virtual printers.

Defining Real Printers

Since all output processed for a given form (that is, virtual printer) is routed to a real printer after the appropriate form has been mounted, care should be taken that no message output interrupts the print process of the queue. It is recommended that a TIBTAB entry for a real printer for which the printout spooling facility is to be used be defined as follows:

```
TIB . . .RMC=1,GROUP=(ALL=NO)
```

Note that the TID should not be defined as a "screen-to-hardcopy" TID.

Printouts can be sent to any VTAM printer without being specified in the Com-plete TIBTAB as long as an unique destination ID (no group ID) is provided during PSOPEN or when using print functions from the Com-plete utilities. In this case TIB entries are acquired dynamically and released after successful printing.

Batch Output Facility (SYSOUT)

This section provides an overview of the extended batch output facilities provided by Com-plete via printout spooling.

During initialization, Com-plete dynamically creates a TIB entry named SYSOUT (if it does not already exist). All printout routed to this terminal will be automatically spooled to the JES/POWER output queue. The output will be separated using the DYNALLOC/SEGMENT facility.

For example, a printout spool destination of SYSOUT=X causes the printout to be routed to JES/POWER in CLASS=X.

Defining JES/POWER Nodes

Defining TIB entries with DEL=YES and ALTTIB=nnn (where *nnn* is the SYSOUT terminal ID) results in routing all printout for these destinations to the JES/POWER queues as described above. However, the NODE name generated for the DYNALLOC/SEGMENT is the TIBNAME of the original terminal. Using this method, printout can be routed directly to any specific JES/POWER node.

Note that setting applymod 85 will cause dynamically generated printers to be allocated with these attributes.

Dynamic Routing

The user exit ULMSBTCH can also influence the JES/POWER routing parameters.

For example, ULMSBTCH could request that printout be spooled to a JES/POWER queue by modifying the parameter ULSTNODE to contain NODE01 and ULSTUSER to contain USER01. Com-plete then issues a DYNALLOC/SEGMENT with the modified parameters and the printout is routed to the specified node/user ID.

Note:

The SEGMENT statement provides a POWER listname as well as a \$\$ LSTcard where the different keyword parameters are supplied as below.

```
* $$ LST DISP=d,CLASS=c,FNO=ffff,DEST=(node,userid)
```

The SEGMENT expression in this description refers to this \$\$ LST card. For more information on the ULMSBTCH exit, see the chapter Security and User Exit Facilities.

Logical Output Drivers

This section explains the logical output driver facility available in Com-plete's printout spooling system.

What is a Logical Output Driver?

A logical output driver is a user-written routine that allows modification, extension and/or translation of output data at the time the printout is routed to the physical printer.

The output driver runs under control of Com-plete's MSGPO task and is loaded dynamically from the COMPLIB dataset(s). It is deleted as soon as its use counter is zero. All operating system functions (OPEN, CLOSE, SVCs, etc.) are available for use. The output driver routine has to be serially reusable since it can be invoked for multiple printouts in parallel.

Note:

The Com-plete functions (MCALL) as described in the Com-plete Application Programming documentation are NOT available to the logical driver facility.

The driver can be used for the following:

- Precede the output data with printer-specific escape sequences;
- Add graphics to the printout;
- Translate pseudo-commands from the application to printer-specific commands.

The source library supplied with the installation tape contains 2 examples of logical output drivers (LDRVSAMP and IPDSDRV).

One of the major benefits in adding printer-specific commands and/or escape sequences using a logical output driver is that when your site changes its hardware, only the output driver routine requires modification: all user applications that create printouts can remain unchanged.

Specifying Logical Output Drivers

The name of the output driver associated with a printout can be supplied via the LDRIV keyword of the PSCB used for PSOPEN processing, and it can be changed or added using the functions of the USPOOL utility. Installations wanting to use a general output driver can use the POLDRV sysparm specification. The driver specified in the POLDRV sysparm will be used for all those printouts which have no driver defined explicitly.

Logical Output Driver Interface

The communication area between the MSGPO task and the output driver is mapped by the CMLDRVW macro from the supplied source library, which is passed via register 1 (R1) on entry. Each output driver must have specified the appropriate definition:

```
name    CMLDRVW TYPE=USER.
```

The interface area consist of a fixed header (DRVBUFF-DRVPARM) followed by a variable area, the size of which is dependent on the number of lines and the linelength specified for this printer:

```
varlength = No.lines * linelength
```

The total length of the passed area is provided in the field DRVPARML. None of the fields in the fixed part of the interface area should be used as working fields for the output driver routine.

This interface area will remain active until the end of the printout, or until the driver returns an end-of-work condition (RC=16), and can therefore be used to keep status information for the individual printout.

The driver routine is invoked for each record in the printout, and the following register settings are passed :

R1 address of the workarea (CMLDRVW TYPE=USER)

R2 COMREG

RD 18 word savearea

RE return address

RF entry point.

The field DRVPARMA contains the address of the current record; the length of this record is in field DRVRECL. These fields are set by Com-plete on entry and must be set by the output driver prior to return to the calling module.

On entry, DRVPARMA points to DRVREC within the communication area. However, the output driver must always use the address passed in DRVRECA and not rely on the contents of DRVREC. The individual output line is passed to the output driver prior to any translation of linefeed and/or formfeed characters in the same format as provided by the application that created the printout. The ASA control characters are converted to the appropriate printer control orders after the output driver is invoked.

Return Codes

Code	Meaning
0	<p>unmodified</p> <p>The line is added to the output buffer unchanged. Com-plete does not check for any values supplied in DRVRECA and DRVRECL.</p>
4	<p>record modified</p> <p>The output driver must provide the address of the modified record in DRVRECA and the length of the new record in DRVRECL. The data is added to the output buffer.</p>
8	<p>insert record</p> <p>The output driver must provide the address of the new record in DRVRECA and its length in DRVRECL. The data is added to the output buffer. The original record from the printout is passed to the output driver again on the next invocation.</p>
12	<p>delete record</p> <p>The current record from the printout is skipped.</p>
16	<p>terminate driver</p> <p>The current record is added unchanged to the output buffer. The work area is freed and the use counter of the driver routine is decreased, and if zero, the driver is deleted from virtual storage. Any remaining records in the printout are passed to the printer as if no logical output driver were specified.</p>
20	<p>send data asis</p> <p>The normal output buffering used for printouts cannot take account of data that has to be sent within a single output request: Com-plete's buffering always adds up output lines until the buffer (the size of which is determined by the line length and number of lines) is full. On the other hand, creating bar codes using Intellegent Printer Data Stream (IPDS) commands requires that all data for a page is passed on a single output request. The data pointed to by DRVRECA with the length contained in DRVRECL will be send to the printer without any further modifications by Com-plete.</p> <p>Same as code 8 (insert record) above, but Com-plete will send the data unmodified, including the first byte (not treating it as an ASA control character).</p> <p>Note that this response has no effect on the positioning within the current printout, that is, the next time the logical output driver is called, the same printout record will be provided. It is the responsibility of the output driver to ignore this record (if required) with a delete record response.</p>

Line Printer Daemon (LPD) Protocol

The LPD Protocol is described under the folliwng headings:

- Introduction

- Concepts
- Printing via the Local Workstation
- Printer Definition Using Environment Variables
- Printer Search Sequence
- LPD Spool vs. Com-plete Spool
- EBCDIC - ASCII Conversion
- Logical Output Drivers
- Printing via Printer "Boxes" Supporting the LPD Protocol
- Changing Printer Definitions Dynamically

Introduction

The LPD protocol is a print server protocol widely used in UNIX networks. In detail, this protocol is described by RFC 1179 which can be found on the Internet, for example at <http://www.faqs.org/rfcs/rfc1179.html>.

LPD server software is available from many different vendors for UNIX and Windows platforms.

Com-plete implements the client part of this protocol, allowing you to route printouts to an LPD print queue instead of the Com-plete spool. From the application programs' point of view, it makes no difference where the printout is routed, the API remains unchanged. This implies that any existing application which create printouts under Com-plete can be used unchanged for printing to LPD print servers. There is one limitation, though: Destination lists with more than one destination are not supported through this interface.

Concepts

An LPD print server usually is a (UNIX or Windows NT) computer with printers connected to it. Each printer is represented on the server by means of one or more print queues. The printer drivers are installed on this server. Different queues for the same printer can be used, for example, to use different fonts or formats like landscape orientation.

From the above follows that it takes two parameters to unequivocally address a printer:

- the IP address of the LPD server, and
- the name of the queue on this server.

In Com-plete, a printer is known by an 8-character name, or a number. Two different ways are available to map these printer names or numbers to LPD printers:

1. You can setup a central table for Com-plete (a list of environment variables) in which you define each of the printer names or numbers to be used in Com-plete, and specify the corresponding LPD server and queue name for it, like `printer=server/queue` (see section Printer Definition Using Environment Variables below).

- In UUTIL UD (personal defaults), every user can specify an LPD server in the field labeled "Server". If this value is set, then every printout created by this user will be routed to this server, and any printer name this user enters in Com-plete will be interpreted as a print queue name on this server.

Printing via the Local Workstation

When a user connects to Com-plete using Com-plete's individual Telnet tn3270 port, then Com-plete knows the current IP address of this user's workstation. If there is an LPD server active on this workstation, then it is possible to route printouts to this LPD server using the current IP address from the Telnet session. This is useful especially in networks with dynamically assigned IP addresses. To make use of this feature, the user must enter an asterisk ("*") in the Server field in UUTIL UD.

Printer Definition Using Environment Variables

The POSIX layer now available in Com-plete supports the concept of environment variables. See *SMARTS Installation and Operations documentation* for details. A convenient way of setting environment variables is to define ENVIRONMENT_VARIABLES=DD:CONFIG in the POSIX parameters (POSIX parameters can be concatenated with the Com-plete SYSPARMS). This tells Com-plete that in your Com-plete JCL procedure there is a statement like this:

```
//CONFIG DD DISP=SHR,DSN=MY.SOURCE(ENVIRON)
```

where MY.SOURCE(ENVIRON) contains the environment variables.

The environment variable setting for an LPD printer must look like this:

```
SAG_APS_LPD_printer=server/queue
```

Where

printer	is the name or number of the printer as used in Com-plete,
server	is the name or IP address of the LPD server, and
queue	is a printer queue on this server. If the queue is omitted, Com-plete uses the default "LPT1".

Example:

```
SAG_APS_LPD_32=my.lpd.server/queue1
SAG_APS_LPD_PRINTER1=my.lpd.server/queuexyz
SAG_APS_LPD_PRINTER2=111.222.333.444
SAG_APS_LPD_PRINTER3=111.222.333.444/lpt3
```

Printer Search Sequence

In general, you can print to both LPD and VTAM printers from the same Com-plete. Com-plete will try to resolve the printer name or number entered in the following order:

- If the user has an LPD server set in his personal defaults, then every printout created by this user will be routed to this server, and any printer name this user enters in Com-plete will be interpreted as a print queue name on this server.

2. If the printer name or number is mapped by means of an environment variable (see section Printer Definition Using Environment Variables), then the server and queue name from this environment variable will be used.
3. Otherwise, the printout will be written to the Com-plete spool, and Com-plete will then asynchronously try to route the printout in the traditional way, using the definitions in the TIBTAB, VTAM, etc.

LPD Spool vs. Com-plete Spool

Since an LPD print server includes a spool of its own, printouts routed to an LPD print server are not spooled in Com-plete. This also means that if the printout cannot be delivered to the server (for example because the queue name specified does not exist), then it is discarded by Com-plete. If the printout cannot be delivered due to a situation that is likely to exist only temporarily, then Com-plete will retry a reasonable number of times before discarding the printout. When Com-plete terminates, all LPD printouts not delivered are lost.

Copying or moving printouts from the Com-plete spool to an LPD print server or vice versa is not supported in this version of Com-plete.

EBCDIC - ASCII Conversion

All data supplied through the API function PSPUT must be in EBCDIC encoding, and is converted to ASCII by Com-plete. Line breaks are converted into CR LF (carriage return + line feed) as expected by most LPD print servers. An ASA control character '1' ("new page") on the first line of a printout is removed, and every printout is terminated by an ASCII FF (form feed). Applmods 71, 72, 87, and 95 are ignored for printouts to LPD print servers.

SCS printer control data is not supported through this interface in this version of Com-plete.

Logical Output Drivers

Logical output drivers are supported for printouts to LPD print servers, however, in most cases the task that used to be solved by a logical output driver can be solved in a more convenient way by simply using different queues on the LPD server.

If this is not possible, then most likely your logical output drivers need to be adopted, because the requirements to these programs are stronger than they used to be for traditional logical output drivers:

1. A logical output driver is automatically loaded as a Com-plete-resident program when called for the first time for a printout being sent to an LPD print server, therefore, it must be re-entrant. Traditional logical output drivers ran only under control of the Com-plete MSGPO task. When called for a printout being sent to an LPD print server, the logical output driver runs in the same task as the program creating the printout. This means, it can be called by more than one program at the same time.
2. Logical output drivers must be able to run in AMODE 31, and return to the calling program in its proper AMODE.
3. Traditional logical output drivers used to insert escape strings designated for mainframe-oriented print servers. These print servers are bypassed when printing via an LPD print server. Therefore, the logical output driver must now supply the original escape strings as expected and understood

by the printer. In addition, you'll have to setup the LPD queue accordingly ("raw mode") so it will let these strings through to the printer unmodified. Note that the logical output driver still must supply EBCDIC data.

Printing via Printer "Boxes" Supporting the LPD Protocol

Many printer network "boxes" support the LPD protocol directly, so you might be tempted to send printouts from Com-plete directly to the printer. This is possible, but doing so you lose the features provided by an LPD server, mainly the possibility to install a printer driver, and the advantages of a spool. Therefore, Software AG recommends that you use real LPD server.

It is unlikely that printer manufacturers will develop printer drivers for Com-plete, so if you want to send printouts directly to the printer, you'll have to provide control information (so called "escape strings") to the printer in order to select a font, formatting, etc. The printer definition environment variables (see above) can be used to provide an escape string to be sent to the printer before the printout, and another one to be sent after the printout:

```
SAG_APS_LPD_printer=server/queue/escape1/escape2
```

If you use this notation, then "escape1" will be sent before each printout, and "escape2" afterwards.

A number of options exists how to define these escape strings:

1. You can enter escape strings directly into the printer definition. Any period (".") will be translated into the ASCII escape character <ESC>.

Example:

```
SAG_APS_LPD_PRINTER1= my.lpd.server/queuexyz/.&a6L.(s0p12H/.E
```

2. You can define escape strings using separate environment variables starting SAG_APS_ESC_ . These variables can then be referenced in printer definitions.

Example:

```
SAG_APS_ESC_myescape=.&a6L.(s0p12H
SAG_APS_LPD_PRINTER1= my.lpd.server/queuexyz/myescape
SAG_APS_LPD_PRINTER2=111.222.333.444//myescape
```

3. You can define default escape strings to be used with every printer whose definition doesn't specify escape strings explicitly. To do so, assign the default escape strings to the following environment variables:

```
SAG_APS_ESC_DEFAULT_START=
SAG_APS_ESC_DEFAULT_END=
```

4. If a starting escape string is effective for a printer (in any of the 3 ways described above), but no ending escape string, then Com-plete automatically sends the escape string <ESC>E, which is the PCL5 "reset".

Changing Printer Definitions Dynamically

The environment variables described above are read once during Com-plete (or POSIX, to be exact) startup. There are plans for the future to provide a global tool for changing the values of environment variables dynamically, without having to restart POSIX, however, this tool was not available yet when this version of Com-plete was released.

As a temporary solution, a utility program UPRTDEF is available in Com-plete, which allows you to change these values, and also to add new variables. Here's a few notes on how to use it:

- Simply edit the data on the screen and press PF5 to apply your changes.
- Overwriting a variable name adds a new variable without deleting the old one.
- Existing variables cannot be deleted physically. To delete a variable logically, clear its value.

Note that this utility operates only on the active settings in the Com-plete address space, it does not update the file where your original definitions are stored. This means, all dynamic changes you make will be lost after Com-plete is restarted, unless you apply the same changes to the file documentation only.