CONTROL DATA CORPORATION
Interoffice Memorandum

DATE : October 1, 1985

TO : ACSD Personnel LOCATION : ARH207

FROM : Janice Netko Feuling LOCATION : ARH207

SUBJECT : Template Control Characters

This memo is meant to elaborate on the documentation of template control characters for defining a message template.

+N[n] Start a new line in the output text indented by n spaces. If n is omitted, the new line is not indented. NOTE: The DI message formatter and the formatter on the 170 used by NPA ignore the '+N' if it occurs as the last text in a template definition. When a message contains multiple templates, then the following is true of the '+N' control character:

1) The first template in a message will automatically begin on a new line due to the prefixing of the status severity level by the message formatter. Thus, a template that begins a message does not normally have a +N at the beginning of a template definition. (Display command responses with headers and columnized data are an example where the template that begins the message has a +N at the beginning of the definition.)

   FOR EXAMPLE:
   If a message is generated by the templates

   ```
   cme$first_line
   {E First line of message.
   ```

   then the result will appear as

   ```
   -- ERROR -- First line of message.
   ```

2) +N's at the end of a template definition are ignored. If you desire a new line then it should be specified by a +N at the beginning of the template definition for the template that you wish to appear on a new line, not at the end of the definition of the previous template.

FOR EXAMPLE:
If a message is generated by the template

```
cme$first_line
{I First line of message.+N

cme$second_line
{I Second line of message.
```

then the result will appear as

```
 -- INFORMATIVE -- First line of message.Second line of message.
```

NOTE that the +N in this example was ignored. Below is the correct way to define the templates.

If a message is generated by the templates

```
cme$first_line
{I First line of message.

cme$second_line
{I +NSecond line of message.
```

then the result will appear as

```
 -- INFORMATIVE -- First line of message.
 Second line of message.
```

3) +N's occuring within the text of a template definition will always result in a new line.

FOR EXAMPLE:
If a message is generated by the template

```
cme$first_line
{I This will result +N in two lines.
```

then the result will appear as

```
This will result
in two lines.
```

+H[N] Insert spaces until, BUT NOT INCLUDING, column n of the current line in the output text is reached (resulting in the text or variable data being placed in column n). If the current position in the output text is past column n, then one space will be inserted. If n is omitted then insert spaces until the next default tab position in the current line of output text is reached. Default tab positions occur every 8 columns.

FOR EXAMPLE:

If a message is generated by the templates

```
cme$columnizing
{E acb+H6efg
```

then the result will appear as

```
columns 1    6
        +----+--
        abc  efg
```

If you have any questions, please see me.

Janice Netko Feuling

?? EJECT ?? { CALLC DOMLOG
This deck provides information on generating CDCNET Log Messages.

The information you are about to provide will be used to generate a CDCNET Log Message Manual and will provide the necessary information needed for generation of CDCNET Alarm Reports and Network Performance Reports.

PROCESS TO DEFINE CDCNET LOG MESSAGES:

1. Obtain a copy of the deck CMECOMT from the CDCNET Source Program Library.
   SES.GETMOD M=CMECOMT B=SOURCEBASE UN=CDNA G=CMECOMT
   This deck contains common message templates which are intended to to be used across the CDCNET product (i.e., to be used across many areas). Any message template which is used by 2 or more areas must be placed in this deck. Common message templates should be used whereever possible. When defining your log message templates and command response templates always check this deck to see if there are already defined templates which you can use. Also if you define any message templates which can be used across multiple areas, place those templates in this deck. Each area is defined by a group code which has been assigned by integration.

2. Obtain a copy of the deck CMELOG from the CDCNET Source Program Library.
   SES.GETMOD M=CMELOG B=SOURCEBASE UN=CDNA G=CMELOG
   This obtains a skeleton deck which can be used to generate a Log Message Definition deck. Below is a description of the information which must be placed in the Log Message Definition deck.

```
{
{ L O G   M E S S A G E   P U R P O S E
{
{         This section gives a short description of the log message
{         and why it is being generated.
{         E X A M P L E:
{
{         Whenever a Routing Information Data Unit (RIDU) is received
{         which does not follow the accepted format, a log message is
{         generated.
{
{ A C T I O N   R E Q U I R E D
{
{         This section specifies the action to be performed by an operator
{         upon receiving the log message.
{         E X A M P L E   C O N T I N U E D:
{
{         No operator action required. The dynamic routing within CDCNET
{         adapts automatically to the failure.
{
{ D E S C R I P T I V E   M E S S A G E
{
{         This section specifies the actual fields of the log message.
```

A log message is made up of a MASK (fixed text) and fields which get inserted into the MASK (variable fields of log message). The variable fields are those which change from one instance to the next whereas the fixed fields remain constant from one instance to the next. The variable fields of the log message are generated by the software module calling log_request. The address passed on the log_request interface is a buffer address. The buffer conta the variable fields of the log message. The MASK is generated by a offline utility called GENMT, and is combined with the variable fields of the log message by an application which resides on the Cyber Host. The Message Template Data Stores reside on the Host where mass storage is available.

The deck METMDU on the CDCNET Source Program Library should be called to obtain the CYBIL definition of the MDU data element type A brief description of each MDU data type is given below. For further detail refer to section 7.0 of the CDNA GDS entitled 'Management'. Section 7.0 describes the format of the data when represented in the MDU format. CDCNET programmers need not concern themselves with this other than to understand the format for debugging purposes. For programming purposes the programmer simply needs to interface to gen_data_field and get_data_field. These procedures make sure the data gets formatted into the MDU format as described in section 7.0 of the GDS. The data presented to gen_data_field must be right justified in the lowest byte within memory (this should not be surprising). Be careful when using packed data!!!

Binary Octet (bin_octet): Binary octets consist of bytes of binary information. The length field passed to gen_data_field indicates the number of bytes. Binary octets are displayed as their equivalent hexidecimal value. A single octet becomes a 2 digit hex ASCII display.

Character String (char_octet): Character string data contains ASCII characters. The length passed to gen_data_field indicates the numbers of ASCII characters.

Binary String (bin_str): A binary string is a contiguous string of bits. The length passed to gen_data_field indicates the number of bits. A binary string is displayed as 0's and 1's.

Unsigned Integer (bin_int): An integer contains a string of binary bits. The length passed to gen_data_field indicates the number of bits in the integer. A data element of type integer gets displayed as its equivalent decimal value.

Signed Integer (bin_sint): An integer contains a string of binary bits. The length passed to gen_data_field indicates the number of bits in the integer. A data element of type integer gets displayed as its equivalent decimal value. The most significant bit contains the sign bit.

Binary Coded Dec (bcd_char): BCD values range from 0 to 9 or 0000(2) to 1001(2) respectively. Each octet contains 2 BCD data elements. The length field passed to gen_data_field indicates the number of BCD elements in the field. A data element of type BCD gets displayed as its equivalent decimal value.

This section should contain a complete description of the log message including MASK and variable fields.
E X A M P L E   C O N T I N U E D:

```
{
{         | M A S K         || L O G _ M E S S A G E _ B U F F E R
{         |-----------------||----------+---------+-----------------------------
{         |    fixed text   ||   type   |value    |         description
{         |-----------------||----------+---------+-----------------------------
{         |See mask1 below  ||none      |         |Field describing error
{         |                 ||          |         |
{         |-----------------||----------+---------+-----------------------------
{         |See mask2 below  ||binary    |1..512   |The Bad RIDU
{         |                 ||octets    |octets   |
{         |-----------------||----------+---------+-----------------------------
{        mask1 - 'Incorrectly formatted Rotuing Information Data Unit receiv
{        mask2 - 'Routing Information Data Unt = '
{
{ T E M P L A T E  I D s
{
{        This section lists the template id common decks containing the
{        template definitions used for the log message. See step 3 below
{        on how to generate message templates.
{        Programmers should contact the responsible analyst to get message
{        template ids assigned for their respective areas.
{        E X A M P L E  C O N T I N U E D:
{
{        RMETEMP
{
{ L O G  M E S S A G E  I D
{
{         This section specifies the log message id. The log message id
{         uniquely identifies the log message.
{         Programmers should contact the responsible analyst to get log
{         message ids assigned for their respective areas.
{         E X A M P L E  C O N T I N U E D:

CONST
  rme_bad_ridu = min_log_message_id + 432;

{ L O G  M E S S A G E  A T T R I B U T E S
{
{        This section describes the attributes of the log message. A log
{        message can be qualified with 1 or more attributes. Below is a
{        list of attributes and associated codes:
{             A        ACCOUNTING
{             HE       HARDWARE ERROR
{             SE       SOFTWARE ERROR
{             S        STATISTICS
{             EL       EVENTS LOG
{             NS       NETWORK SECURITY
{             INSTALLATION DEFINED TYPES
{        INSTALLATION DEFINED TYPES are new attribute names which the
{        site can assign if the customer does not agree with the CDC
{        defined attributes.
{        E X A M P L E  C O N T I N U E D:
{
{             The example below has assigned the log message to have
{             the attributes of Software Error and Events Log. The
{             The attribute definition is specified in a form directly
{             readable by DADR (an NPA preprocessor), since it is
{             DADR which will read this text.
```

```
{
{SE, EL
{
```

3. Add a '*callc XXELOG' to the CMCLOG deck. The CMCLOG deck can be compiled to produce a listing containing all of the log messages currently defined in the system.

4. Obtain a copy of the deck CMETEMP from the CDCNET Source Program Library.
   SES.GETMOD M=CMETEMP B=SOURCEBASE UN=CDNA G=CMETEMP
   This deck is a skeleton deck which can be used to define message templates for a particular area. XX is the group id for the area as defined by integration.
   Below is a description of the data which must be included in the Template Definition Deck and instructions on how the definitions are made.

```
{ S E V E R I T Y  a n d  L O G  M E S S A G E  T E M P L A T E
{ D E F I N I T I O N
{
{         The severity level of the log message and the text used to define
{         the message template(s)  (i.e., mask) for the log message is
{         specified in the deck XXETEMP.
{
{         A single log message may use several message templates.
{         The first display line of a log message should always describe
{         the condition. The severity level for the log message applies
{         to the severity of the condition.
{         If a log message is made up with the use of 2 or more
{         message templates the first template (which contains the
{         condition description), will be defined with the severity level
{         which will apply to the entire log message. Additional templates
{         used for forming the log message should be defined with a
{         severity level of INFORMATIVE.
{
{         The definition is given in a form directly readable by GENMT,
{         a utility which is used to generate a compilable module from the
{         specified definition. The module is then compiled to produce
{         the desired object module containing the message template.
{         The produced module will be used by the Cyber Host to format the
{         message into an Alarm, or by DADR to produce an NPA Software Error
{         or Hardware Error report.
{         The template definition must be in the following format:
{             template id;
{             {<severity>  <message text (upto 80 characters)>}
{             {<message text continued (if necessary)>}
{
{         Below is a list of the control sequences provided to the CDCNET
{         programmer for DEFINING A MESSAGE TEMPLATE FOR A LOG MESSAGE.
{
{         The plus character (+) is used as a delimiter to indicate
{         control information.   This control character and the
{         characters  immediately  following it  represent actions
{         the  message formatter will perform with the text. In the
{         following,  [n]  is used  to  indicate  that n is optional.
{         n  is defined as  an unsigned  decimal number between 1
{         and 128.
```

+P[n] The nth variable field is inserted in the output text. If n is omitted, n is assumed to be one more than the value used in the previous call. If there has not been a previous call, the first variable field is inserted.

Assuming the following variable field values:
   '111', '222', '333', '444',
'abc+Pdef+P3ghi+Pjkl+P2' will cause
"abc111def333ghi444jkl222" to appear in the output text.
'abc+P-123' will cause
"abc111123" to appear in the output text.

+N[n] Start a new line in the output text indented by n spaces. If n is omitted, the new line is not indented. NOTE: The DI message formatter and the formatter on the 170 used by NPA ignore the '+N' if it occurs as the last text in a template definition. When a message contains multiple templates, then the following is true of the '+N' control character:

1) The first template in a message will automatically begin on a new line due to the prefixing of the status severity level by the message formatter. Thus, a template that begins a message does not normally have a +N at the beginning of a template definition. (Display command responses with headers and columnized data are an example where the template that begins the message has a +N at the beginning of the definition.)

   FOR EXAMPLE:
   If a message is generated by the templates

```
     cme$first_line
     {E First line of message.
```

   then the result will appear as

```
     -- ERROR -- First line of message.
```

2) +N's should not be specified at the end of a template definition. If you desire a new line then it should be specified by a +N at the beginning of the template definition for the template that you wish to appear on a new line, not at the end of the definition of the previous template.

   FOR EXAMPLE:
   If a message is generated by the templates

```
     cme$first_line
     {I First line of message.

     cme$second_line
     {I +NSecond line of message.
```

then the result will appear as

```
-- INFORMATIVE -- First line of message.
Second line of message.
```

3) +N's occuring within the text of a template definition will always result in a new line.

FOR EXAMPLE:
If a message is generated by the template

```
cme$indented_new_lines
{I +Nnow+N2is+N4the+Ntime
```

then the result will appear as

```
-- INFORMATIVE --
now
  is
    the
time
```

+X[n] Insert n spaces in the output text. If n is omitted, 1 space is inserted.

'abc+X5def' will cause "abc   def" to appear in the output text.

++ Insert a single "+" in the output text.

'abcd++efg' will cause "abcd+efg" to appear in the generated text.

+- This option inserts nothing in the output text. It is used to separate control characters from text characters where there can be a conflict.

'abc+X1+-1xx' will cause "abc 1xx" to appear in the generated text, where as 'abc+X11xx' will cause "abc   xx" to appear in the generated text.

+H[N] Insert spaces until, BUT NOT INCLUDING, column n of the current line in the output text is reached (resulting in the text or variable data being placed in column n). If the current position in the output text is past column n, then one space will be inserted. If n is omitted then insert spaces until the next default tab position in the current line of output text is reached. Default tab positions occur every 8 columns.

FOR EXAMPLE:

If a message is generated by the templates

```
'abc+H6efg'
```

then the result will appear as

```
columns 1    6
        +----+--
        abc  efg
```

+R Begin repeating information. The rest of the message text is assumed repeated indefinitely until all the delimited sequences from the text field of the status record are exhausted. NOTE: If n is specified for a P control sequence within the repeating information, the information will be repeated an infinite number of times (i.e., infinite loop -- be careful not to have n specified on the P parameter).

Assume the following variable field values:
'four', 'score', 'and', 'seven', 'years', 'ago'

```
'+P+R+H+P' will produce:
"four    score   and     seven   years   ago"

'+R+P+H+P+H+P+N' will produce:
"four    score   and
 seven   years   ago"
```

If the character following the control character is not one of those quoted above, the results are undefined.

Below is list of the SEVERITY LEVELs which can be assigned to a log message, and the associated codes to be used:

'I'	INFORMATIVE CONDITION
'W'	WARNING CONDITION
'E'	ERROR CONDITION
'F'	FATAL CONDITION
'C'	CATASTROPHIC CONDITION

CDCNET SEVERITY LEVEL DEFINITIONs

INFORMATIVE CONDITION - These messages convey items of general interest and are not a result of incorrect or incomplete operation.
Used for Statistical type log messages, system event log messages (e.g., logging of operator activity -- including all commands entered, responses received, and alarms received), periodic reporting of system configuration information, accounting type information, etc.

WARNING CONDITION - These messages convey items of general interest and may have been the result of incorrect or incomplete operation. Warning indicates that the system is approaching some error condition (i.e., threshold condition).
Used for log messages containing information warning of system resource degradation (e.g., used by Executive to report availabilty of buffers, memory, etc.; has degraded below the acceptable threshold), etc.

ERROR CONDITION - These messages convey that the operation was not completed correctly.

```
{                    A log message is qualified as severity level error if
{                    the message is the result of an error condition which
{                    is correctable by the DI software (i.e., error has
{                    minimal impact on system operation and performance).
{                    Used for parameter verification errors (e.g., if a
{                    communication layer or network management entity
{                    receives invalid parameters at its user interface, the
{                    error is logged by the layer and the user is notified
{                    of the error). This level should be used when reporting
{                    the receiption of bad PDUs. This severity level should
{                    also be used in the case that an action is not allowed
{                    at the time of request.
{               FATAL CONDITION - These messages convey that the operation
{                    was not completed correctly.
{                    A log message is qualified as severity level fatal
{                    if the message is the result of an error condition
{                    which affects the operation of a major portion of the
{                    system but was recoverable by the DI software.
{                    Used to log fatal hardware failures (e.g., used by
{                    lower layer failure management to log device failures),
{                    and fatal software conditions (e.g., used by System
{                    Ancestor to report the occurence of a task failure).
{               CATASTROPHIC CONDITION - These messages convey that the
{                    operation did not complete correctly and resulted
{                    in the least desired recovery.
{                    A log message is qualified as severity level catastrophi
{                    if the message is the result of an error condition which
{                    has severe impact on system/network operation and
{                    performance. Used for severe hardware failures which
{                    affect a large portion of the system (used by lower laye
{                    failure management to log severe device failures). Used
{                    by System Ancestor to indicate that the system required
{                    reloading due to numerous task failures.
{                    Catastrophic means that the DI could not recover without
{                    reload.
{
{          E X A M P L E   C O N T I N U E D:
{               The following statement defines the template id, severity
{               level, and message template to be used for the log message.
{               Note that the 3rd and 4th characters of the template id
{               must be 'e$'.
{

CONST
  rme$rme_bad_ridu = cme$min_template_id + 21;
{E Incorrectly formatted Routing Information Data Unit received}
{+NRouting Information Data Unit = +P1}
{|                                                             |
{|                                                             |
{|-------------------------+-----------------------------------|
{                          |
{                    MESSAGE TEMPLATE
{
{
{ O P E R A T O R   D I S P L A Y   E X A M P L E
{          This shows how the message would appear if it were displayed
{          as an alarm at the operators console.
{
```

```
{          CDCNET ALARM ******
{          system_name  83/08/04  11.00.35  30432
{          --ERROR--  Incorrectly formatted Routing Information Data Unit
{          Routing Information Data Unit = ffedc12450cdcd120123ccf
{
```

5. Add a '*callc XXETEMP' to the CMCTEMP deck. The CMCTEMP deck can be complied to produce a listing containing all of the message templates currently defined in the system.

6. When the module which uses the defined log messages is transmitted the Log Message Definition deck XXELOG and the Template Definition deck XXETEMP should be transmitted under the same PSR. XX is the appropriate group code for the area.
   EXAMPLE:

   There are 4 different log messages which are logged from within the ROUTING M-E. Therefore when the ROUTING M-E is transmitted to I&E, there will be a deck called RMELOG which contains the information for all the Routing M-E log messages, which will also be transmitted. The deck RMETEMP will also be transmitted.

7. HOW TO LOG A MESSAGE IN CDCNET

```
{The xxelog deck containing the log definitions for the particular
{area, the xxetemp deck containing the message template definitions,
{and the XREF decks for gen_data_field, gen_template_id, and
{log_request procedures need to be called into the module.
*callc rmelog
*callc rmetemp
*callc mexgdf
*callc csxgti
*callc lsxlogr

log_msg_bufptr := NIL;
gen_template_id (log_msg_bufptr, rme$rme_bad_ridu);
gen_data_field (log_msg_bufptr, ^ridu, ridu_size, bin_octet);
log_request (rme_bad_ridu, log_msg_bufptr);
```

Additional Background Information:

The template definition lines in the Log Message Definition decks will be read as input to GENMT (Message Template Generator) generate a message template. CDCNET programmers do not need to actually generate the resultant template by running the template definitions into GENMT. CDCNET programmers must simply provide the information requested for in the log definition deck XXELOG and and message template definition deck XXETEMP.

```
?? EJECT ??   { CALLC CMETEMP

?? NEWTITLE := 'CDCNET Template Identifier Range' ??
?? PUSH (LISTEXT := OFF) ??
{ cmetmpr - CDCNET Template Identifier Range
?? POP ??

  CONST
    cme$min_template_id = 0,
    cme$max_template_id = 65535;

  TYPE
    template_id_type = cme$min_template_id .. cme$max_template_id;


?? OLDTITLE ??

?? NEWTITLE :=
    'XXETEMP - id .. id, <feature name> Message Template Definitions', EJECT
?? PUSH (LISTEXT := OFF) ??
{ XXETEMP - Message Template Definitions for <feature name>
?? POP ??
?? FMT (FORMAT := OFF) ??

CONST
  <template id> = cme$min_template_id + <assigned_template_id>;
  {<severity>  <message text (upto 80 characters)>}
  {<message text continued (if necessary)>}

CONST
  <template id> = cme$min_template_id + <assigned_template_id>;
  {<severity>  <message text (upto 80 characters)>}
  {<message text continued (if necessary)>}


?? FMT (FORMAT := ON) ??
?? OLDTITLE ??
?? EJECT ??   { CALLC CMELOG

?? NEWTITLE := 'CMEECCR - CDCNET Exception Condition Code Ranges' ??
?? PUSH (LISTEXT := OFF) ??
{ CMEECCR - CDCNET Exception Condition Code Ranges
?? POP ??


CONST
  min_log_message_id = 0,
  max_log_message_id = 32999,

  min_response_message_id = 33000,
  max_response_message_id = 65535;

TYPE
  log_msg_id_type = min_log_message_id .. max_log_message_id;
?? OLDTITLE ??

?? NEWTITLE := 'XXELOG : id .. id, XX Log Message Definitions', EJECT ??
?? PUSH (LISTEXT := OFF) ??
{ XXELOG  - XX Log Message Definitions
?? POP ??
?? FMT (FORMAT := OFF) ??

  { L O G   M E S S A G E   P U R P O S E
  {
  {
  { A C T I O N   R E Q U I R E D
  {
  {
  { D E S C R I P T I V E   M E S S A G E
  {
  {        | M A S K          || L O G _ M E S S A G E _ B U F F E R
  {        |------------------||---------+---------+--------------------------
  {        |    fixed text    ||  type   |value    |        description
  {        |------------------||---------+---------+--------------------------
  {        |                  ||         |         |
  {        |                  ||         |         |
  {        |------------------||---------+---------+--------------------------
  {        |                  ||         |         |
  {        |                  ||         |         |
  {        |------------------||---------+---------+--------------------------
  {        |                  ||         |         |
  {        |                  ||         |         |
  {        |------------------||---------+---------+--------------------------
  {
  { T E M P L A T E   I D s
  {
  {        <list of common decks containing template definitions>
  {
  {
  { L O G   M E S S A G E   ID  and  L O G   M E S S A G E   A T T R I B U T E S

CONST
  <log message id> = min_log_message_id + <assigned_log_message_id>;
  {<list of assigned attributes>
```

```
?? EJECT ??

?? FMT (FORMAT := ON) ??
?? OLDTITLE ??
```