Communication Server 2100

ASCII SMDR Data Access Description and Implementation

555-4001-119

Document status: Standard
Document version: 07.01
Document date: 20 October 2006

Contents

New in this release

There have been no updates to the document in this release.

About this document

Purpose and Audience

The SMDR Data Access feature permits SMDR data transmission from a Meridian SL-100 system to either a collocated or remote Down Stream Processor (DSP). The system accomplishes this transmission via the Maintenance and Administration Position (MAP) commands that initialize the datalinks and initiate a one-way flow of data across them.

How to check the version and issue of this document

The version and issue of the document are indicated by numbers (for example, 01.01.)

The first two digits indicate the version. The version number increases each time the document is updated to support a new software release. For example, the first release of a document is 01.01. In the next software release cycle, the first release of the same document is 02.01.

The second two digits indicate the issue. The issue number increases each time the document is revised but re-released in the same software release cycle. For example, the second release of a document in the same software release cycle is 01.02.

ATTENTION

FOR MORE INFORMATION

To determine whether you have the latest version of this document and how documentation for your product is organized, check the release information in the *Meridian SL-100 Master Index of Publications*.

References in this document

The following documents are referred to in this document:

- *Meridian Digital Centrex Station Message Detail Recording Reference Manual*
- *Customer Data Schema Reference Manual*

Introduction to station message detail recording

Purpose

The Station Message Detail Recording (SMDR) Data Access feature provides data transfer of SMDR information from the Meridian SL-100 system to any Down Stream Processor (DSP) that can support the necessary protocol. It offers an alternative way of obtaining call record information, besides tape and disk files. The result is instantaneous and the process is not restricted by start and stop events, as with using tapes and disks.

SMDR performs the following functions:

- Defines one or more links between the Meridian SL-100 system and the DSP designated for SMDR data flow
- Obtains formatted SMDR records at the same time as handing them to the Device Independent Recording Package (DIRP) system
- Formats the SMDR records into messages recognized by the multilink ASCII device driver interface
- Sends these messages to the multilink ASCII device driver interface, which in turn transfers the messages to the DSP

The SMDR Data Access feature is also used in conjunction with the Automatic Call Distribution (ACD) feature packages as a means of reporting statistics related to the ACD operation via a 2-way data transportation data link. The data link must be datafilled in Table TERMDEV like any other terminal device, and the data link characteristics must be defined in Table SLLNKDEV. The link can be made ready by the appropriate commands provided in the LNKUTIL Command Interpreter (CI) increment and the CI increment, SMDRLNK. A message displays, which indicates the success or failure of the command request.

This feature is limited to the generation of SMDR call records. Records are sent as messages, in the format received, to the multilink ASCII device driver interface and are transferred to the DSP.

Data transfer for SMDR is compatible with ACD management reports, in that they can reach the DSP by the same data link. They coexist, but are functionally independent. Commands within their respective CI increments regulate which reports are affected: SMDRLNK for SMDR reports and ACDPOOLS for ACD management reports.

Two log reports, SLNK102 and SLNK103, are generated for each data link device whenever data transfer starts or stops, respectively.

Limitations to SMDR message flow

Data link overflow can occur if the links are unable to handle the amount of data being transferred. If overflow occurs, the oldest call records are discarded to make room for current information, and the system generates logs to inform the user of records loss and link overflow condition.

Any type of switch restart stops link transmission and message generation. All pending messages are destroyed by the restart. Once the link is re-established, SMDR data transfer must be manually restarted.

SL-100 Hardware requirements for SMDR

One NT1X67 terminal controller card must be provisioned for each data link. Two versions of the NT1X67 terminal controller card are available: NT1X67BC (4800 baud) and NT1X67EA (9600 baud).

Due to message flow limitations, only one port per NT1X67EA card should be datafilled. Up to four ports can be datafilled on the NT1X67BC card, depending on the baud rate used. At 4800 baud (maximum baud rate), only one port per NT1X67BC card can be datafilled. Lower baud rate applications enable more ports to be datafilled.

One NTFX30 terminal controller port must be provisioned for each data link. Each IOM port can support speeds up to 19.2K baud. For additional transfer capacity, up to four IOM ports can be provisioned for one ASCII SMDR link.

One Ethernet Interface Unit (EIU) can be provisioned to transport the ASCII data from the Meridian SL-100 over an Ethernet LAN to the billing processor.

A Maintenance and Administrative Position (MAP) terminal must be available to establish the data links and initiate data flow across the links.

A customer-provided down stream processor must be available to process SMDR call records sent by the Meridian SL-100 system.

SL-100 Hardware requirements for SMDR

One NT1X67 terminal controller card must be provisioned for each data link. Two versions of the NT1X67 terminal controller card are available: NT1X67BC (4800 baud) and NT1X67EA (9600 baud).

CS 2100 Compact Hardware requirements for SMDR

On pure IP systems, the Ethernet interface on the Call Server delivers the ASCII SMDR data to the CS-LAN. For support of a RS-232 interface, a terminal server can be used. The NTVW02GA In-Reach Terminal Server can provide this function.

Data link protocol definition

Introduction

This chapter describes the protocol used to provide ASCII Station Message Detail Recording (SMDR) call records to an RS-232-C or Ethernet data link on the Meridian SL-100 system. This protocol is compatible with the International Telegraph and Telephone Consultative Committee (CCITT) X.409 and X.410 recommendations wherever possible.

The protocol used for this feature reduces incremental development on behalf of the receiver of the SMDR information when this system is upgraded. As such, its design maximizes compatibility with existing protocol development directions, as well as consistency with concurrent protocol design work.

This section also includes the information to process the messages at the receiving end of the interconnecting link.

In general this chapter describes the following topics:

- Protocol background - X.409 and X.410 protocol definition recommendations and characteristics of Open Systems Interconnection (OSI) protocols
- SMDR protocol events - the major messages
- Message contents - contents of the messages
- Examples - examples of all of the messages
- Message sequences - message sequences and recovery from failure and restart conditions

Protocol layers

The SMDR Data Access feature uses the following protocols for the seven layers of the OSI model:

- Physical - RS-232-C
- Datalink - ASCII information. Only ASCII alphanumerics are transmitted. The system transmits the unidirectional data link to the DSP.
- Network - IP

- Transport - TCP. ASCII line-feed and carriage-return characters, at both the beginning and end of each packet/message, separate the packets.
- Session - none
- Presentation - X.409 presentation syntax
- Application - SMDR data access protocol using X.410 remote operations

Specific association control message pairs, which establish communication across a link, initiate and terminate interapplication communication across specified links. Other message pairs, in turn, initiate and terminate the transfer of SMDR information.

X.409/X.410 protocol definition

This protocol is defined based on the remote operation concept embodied in CCITT recommendation X.410 and the presentation layer transfer syntax and formal notation defined in CCITT Recommendation X.409. This definition provides consistency for any changes that might take place to the protocol or in some of the uses of the SMDR data access software. This definition also enables more than one function or application to use the same links.

X.410 remote operations

A Remote Operation (RO) is the fundamental unit of operation for an application layer protocol defined within the constraints of Recommendation X.410. An RO can be viewed as a request for processing at a remote location. The RO contains both the request and the reply.

An RO is the exchange of messages, called Operation Protocol Data Units (OPDUs), intended to arrive at a specific result.

The X.410 recommendation defines the following four classes of OPDUs:

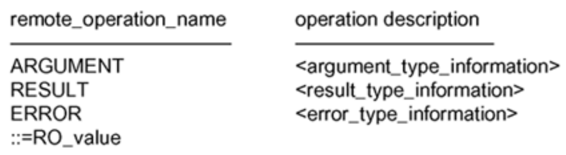
- Invoke OPDU - initiates the activity
- ReturnResult OPDU - returns the result of the invoke operation
- ReturnError OPDU - returns an error indication to the invoker
- Reject OPDU - indicates an invalid or unknown invoke or other message

All invokes specify the operation to be invoked and any parameters to be associated with the invocation. The ReturnResult OPDU contains the values to be returned from the operation.

The SMDR Data Access feature does not use the ReturnResult OPDU, ReturnError OPDU, or Reject OPDU. The nature of the data link enables it to return parameters for a meaningful result.

Remote operation definition syntax

The following syntax describes an RO:



remote_operation_name	operation description
ARGUMENT	<argument_type_information>
RESULT	<result_type_information>
ERROR	<error_type_information>
::=RO_value	

The remote_operation_name is the title of the operation. The argument_type_information describes the information used for the Invoke OPDU. Result_type_information describes the information associated with the ReturnResult OPDU. The error_type_information lists the types of errors that are acceptable as values for the ReturnError OPDU. RO_value is the value given to distinguish this RO from all others.

The following types of information can be presented as argument, result, or error:

- Basic - integer, octet string, bit string, boolean, or numbers within a specified range
- Composite - sequences, sets, or choices of basic or composite types
- Named - named representations of the preceding types

X.409 description

Recommendation X.409 is an emerging international standard that is a method for encoding information that allows the passing of type information along with the contents of the described type. For example, rather than just passing an integer value (such as 14355), the type (INTEGER) and length (in bytes) are also transmitted, allowing the receiver of the message to interpret the value correctly. This adds some overhead to the messages, but also allows message formats to be defined that are more flexible and dynamic. It also gives the receiver of a message the opportunity to interpret a message without exact prior knowledge of a fixed format.

The X.409 Recommendation contains two components: formal notation for protocol specification and syntax for message transfer.

X.409 characteristics

X.409 is based on the concept of a data element. A data element contains the following parts:

- Identifier
- Length
- Contents

Identifier

The identifier part of a data element is the information that allows the data element to be decoded. The identifier is comprised of a class, a form, and an ID part.

The identifier class contains one of the following types of information:

- Universal - pre-defined types of information such as INTEGER, BOOLEAN, OCTET STRING, NULL, etc.
- Application-wide - defined the same for all instances within a given application
- Context-specific - definition of data elements differ from use to use within an application and are interpreted from the context in which they are found
- Private - for private use

The identifier form contains one of the following types of information:

- Primitive - atomic values such as INTEGER
- Constructor - collections of other data elements

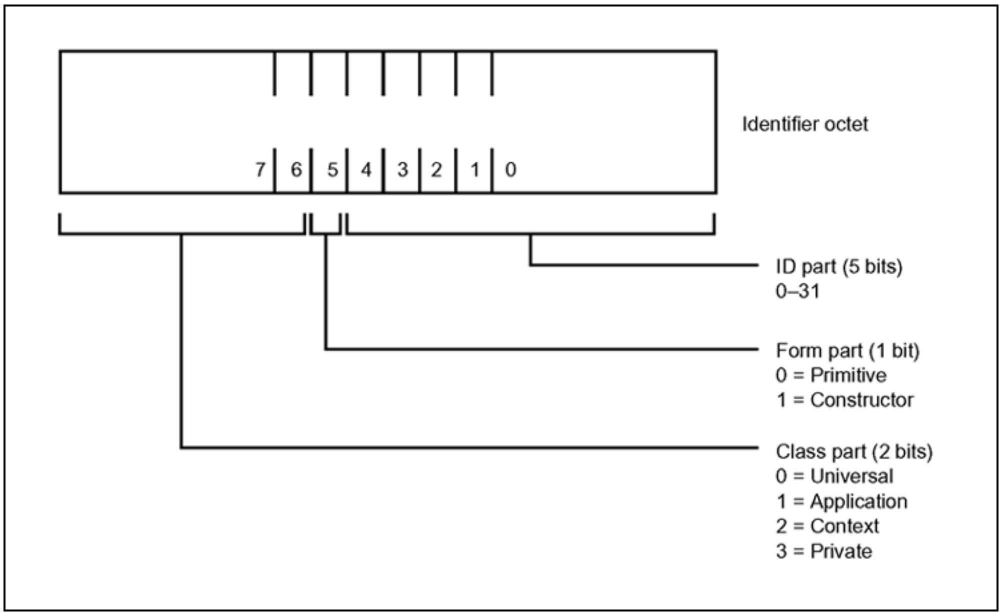
The identifier ID specifies which particular identifier is intended. The ID parts for the universal class identifiers INTEGER and BOOLEAN are different, allowing the two data elements to be distinguished. ID part values can be shared between classes. For example, the identifier, CLASS = UNIVERSAL and ID = 10, is different from the identifier, CLASS = CONTEXT-SPECIFIC and ID = 10.

The identifier of a data element occupies a byte and is constructed as shown in Figure 1 "Data element identifier" (page 17).

Length

The length part of a data element distinguishes the number of bytes consumed in the contents part of the data element. If the contents part contains other data elements, then the length should be the sum of the actual lengths of the contained data elements.

Figure 1
Data element identifier



Contents

The content part of the data element is either the value, if the form type is primitive, or a collection of data elements, if the form type is constructor.

By repeated application of this rule, the data element can be resolved into a collection of primitive data elements, and thus into a collection of values.

Table 1 "Universal class identifier ID values" (page 17) shows some of the ID values for universal class identifiers.

Table 1
Universal class identifier ID values

Identifier	Value
Integer	#02
Boolean	#01
Null	#05
Sequence	#10
Set	#11
A5String	#16
Note: In a sequence and a set, the form is "1" because they are constructor (complex) types of information.	

This is only a partial list of universal class identifier bytes. It does not address application, context, or private class identifiers.

Example

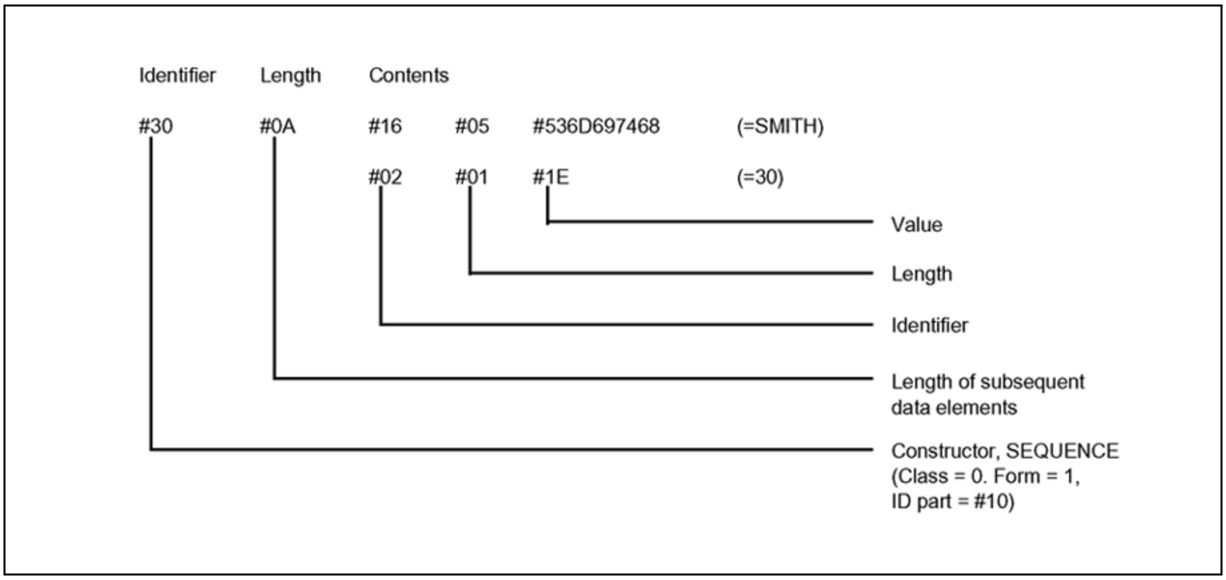
Table 2 "Encoding an integer value" (page 18) shows an example of encoding the integer value, 828343 = #CA367.

Table 2
Encoding an integer value

Identifier	Length	Value
#02	#03	#CA367

The encoding of a name and an age such as SMITH and 30 would be a constructor encoded as shown in Figure 2 "Encoding a constructor" (page 18).

Figure 2
Encoding a constructor

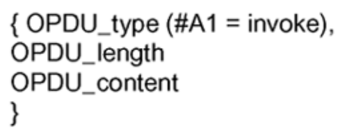

Identifier
Length
Contents
#30
#0A
#16
#05
#536D697468
(=SMITH)
#02
#01
#1E
(=30)
Value
Length
Identifier
Length of subsequent data elements
Constructor, SEQUENCE (Class = 0. Form = 1, ID part = #10)

Encoding guidelines

The encoding of Invoke OPDU of the SMDR-related RO follows the guidelines established in the preceding paragraphs on X.409. Each invoke can be considered as a message that is dispatched intending to achieve a specific purpose.

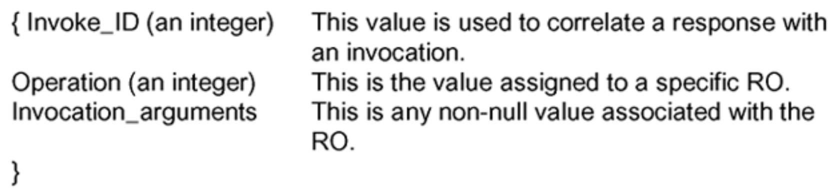
Invoke OPDU has an OPDU type of #A1 and a length byte that is the length of the entire OPDU.

All invocations take the following form:



```
{ OPDU_type (#A1 = invoke),
OPDU_length
OPDU_content
}
```

The OPDU_contents are a sequence of:



```
{ Invoke_ID (an integer)     This value is used to correlate a response with
                             an invocation.
Operation (an integer)       This is the value assigned to a specific RO.
Invocation_arguments         This is any non-null value associated with the
                             RO.
}
```

This invocation is expressed as shown in Figure Figure 3 "Encoding OPDU" (page 20). All values are expressed in hexadecimal unless mentioned otherwise.

SMDR remote operations

The SMDR RO is described in the paragraphs that follow, on a one-by-one basis, using the tools provided by X.410.

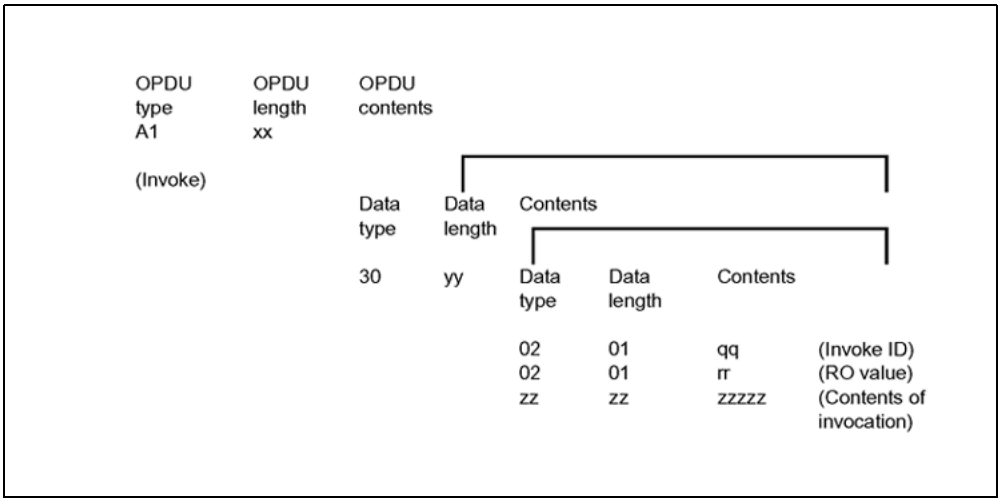
Message content

The message content is encoded using the recommendations put forth in CCITT X.409, which deals with encoding data for messaging. For a complete description, refer to CCITT Recommendation X.409.

ASCII conversion

So that the protocol specification is independent of the transmission scheme used across the link(s), all of the values in the messages are described using bytes or octets.

Figure 3
Encoding OPDU



In the first instance of this protocol, the link used is capable of transmitting only ASCII characters, so all messages are converted from bytes to ASCII characters. At both the beginning and end of each message, the messages are separated by ASCII line-feed and carriage-return characters. Each nibble of the bytes is converted into ASCII representations of that nibble.

The translation is as follows:

Table 3
Byte to ASCII translation

Nibble	ASCII
0	30 = "0"
1	31 = "1"
2	32 = "2"
3	33 = "3"
4	34 = "4"
5	35 = "5"
6	36 = "6"
7	37 = "7"
8	38 = "8"
9	39 = "9"
A	41 = "A"
B	42 = "B"
C	43 = "C"

Nibble	ASCII
D	44 = "D"
E	45 = "E"
F	46 = "F"

Send SMDR call record

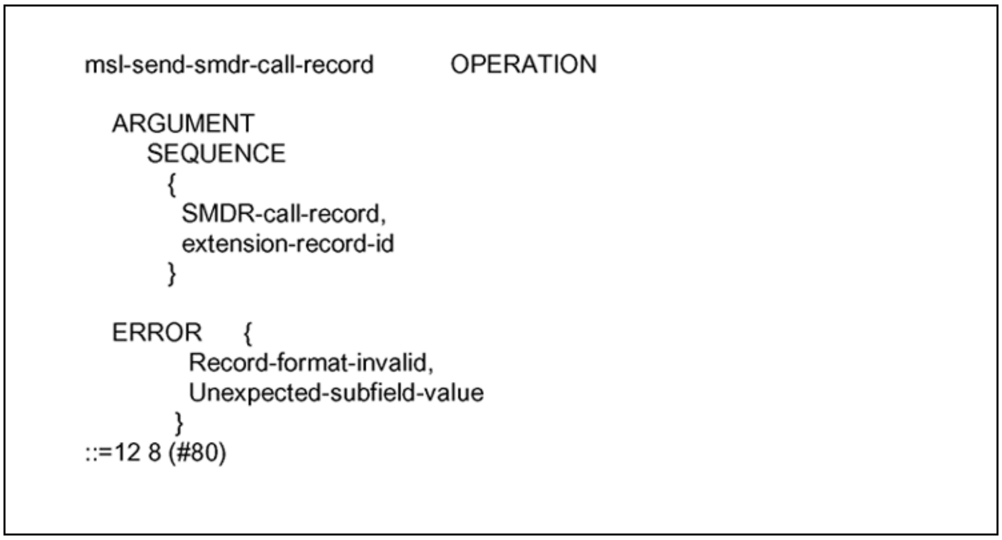
The msl-send-smdr-call-record RO is used to transmit the call information from the Meridian SL-100 system. See Figure 4 "Send SMDR call record" (page 21).

The extension_record_id appears in an msl_send_smdr_call_record RO if there are extension records that are to be associated with the current SMDR call record. These extension records are contained in another RO.

This ID appears in both the main call record (D1-D4) and in any subsequent RO containing extension records. The same value of the ID is contained in all ROs that pertain to the same call.

Due to the fact that the link used is unidirectional, the RESULT and ERROR OPDU are currently nonexistent. With a bidirectional link capability, the only possible value returned as a result of a record is an error indication implying message errors or unexpected (but potentially valid) SMDR sub-record field values.

Figure 4
Send SMDR call record



```
msl-send-smdr-call-record        OPERATION

   ARGUMENT
      SEQUENCE
        {
          SMDR-call-record,
          extension-record-id
        }

   ERROR    {
              Record-format-invalid,
              Unexpected-subfield-value
            }
::=12 8 (#80)
```

SMDR call types

The smdr-call-record is defined as follows:

```
smdr-call-record    ::=
    SEQUENCE OF CHOICE {
```

Table 4
Short-form-D1-SMDR-record [0] IMPLICIT OCTET STRING

Byte	Field
0-1	cust_group_and_origtype
2-7	orig_id
8	info_digs
9	console_number
10	subgroup_and_termtype
11-16	term_id
17-18	route_info_and_date
19-21	time_of_day
22-24	elapsed_time
25	feature_code
26-31	short_called_digits

Table 5
Short-form-D2-NERVE record [1] IMPLICIT OCTET STRING

Byte	Field
0-1	cust_group_and_origtype
2-7	orig_id
8	info_digs
9	console_number
10	subgroup_and_termtype
11-16	term_id
17-18	route_info_and_date
19-21	time_of_day
22-24	elapsed_time

Byte	Field
25	feature_code
26-31	short_called_digits

Table 6
Long-form-D3-SMDR record [2] IMPLICIT OCTET STRING

Byte	Field
0-1	cust_group_and_origtype
2-7	orig_id
8	info_digs
9	console_number
10	subgroup_and_termtype
11-16	term_id
17-18	route_info_and_date
19-21	time_of_day
22-24	elapsed_time
25	feature_code
26-40	long_called_digits

Table 7
Long-form-D4-NERVE record [3] IMPLICIT OCTET STRING

Byte	Field
0-1	cust_group_and_origtype
2-7	orig_id
8	info_digs
9	console_number
10	subgroup_and_termtype
11-16	term_id
17-18	route_info_and_date
19-21	time_of_day
22-24	elapsed_time

Byte	Field
25	feature_code
26-40	long_called_digits

Table 8
Digits As Outpulsed (DAO)-extension-D5-record [4] IMPLICIT OCTET STRING

Byte	Field
0 -14	outpulsed_digits
14.5	digits_missing

Table 9
Auth-Acct-code-D6-record [5] IMPLICIT OCTET STRING

Byte	Field
0	record_type
1-7	auth_or_acct_code

An smdr-call-record is really zero or more octet strings, each representing one SMDR record.

Each RO only contains octet strings that pertain to a single call.

The extension-record-id is defined to be:

```
extension-record-id   ::=
           SEQUENCE
                          {
                          extension-id
                          extension-contents
                          }
where
           extension-contents    ::= INTEGER {
                     (0) no-extension-needed,
                     (1) auth-acct-extension-needed,
                     (2) DAO-extension-needed,
                     (3) both-extensions-needed
                                                    }
extension-id                          ::= INTEGER
```

The extension-contents specify what extension information is needed to complete the call. If, for example, the extension-contents value of 3 (both-extensions-needed) is present in an msl-send-smdr-call-record RO, then the call data is not complete until RO containing DAO-extension-D5-records and auth-acct-code-D6-extension records has been received for the same call.

Note that both the DAO and auth-acct-code extension records might be contained in the same RO or might be contained in a separate RO. This means that the number of RO is not indicated by the extension-contents; it simply indicates what information is provided in the extension records.

The preceding record definition refers to a number of types. These types correspond directly to the data format presented in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119, and are not covered in detail in this document. In the event that the type is not covered in detail, the meaning and values are identical to the description in the *Meridian Digital Centrex Station Message Detail Recording Guide*, 297-2071-119.

Table 10
SMDR-record containing the Meridian SuperNode Extension Record, DD

Byte	Field
0-1	cust_group_number
2	orig_type (trunk)
4-6	trunk_group_ID
7	spare
8-9	trunk_member_ID
10-12	spare
13	answer_type (electrical)
14-15	route_information_digits
16-17	console_number
18	subgroup
19	term_type (station)
20-29	station_billing_DN
30-31	spare
32	route_information_digit
33-35	start_time (day)
36	start_time (hour)
37	start_time (minute)
38	start_time (second)
39-45	elapsed_time (seconds)
46	originating_feature_code
47	terminating_feature_code
48-59	called_digits

The following is the Bearer Capability SMDR extension record:

- DD - Meridian Supernode SMDR extension record code
- A9 - Bearer capability
- A6 - Bandwidth of the call

Digits

In any place where digits are presented (as in short_called_digits, long_called_digits, auth_or_acct_code, outpulsed_digits, and occasionally orig_id and term_id), they are presented in sequential bytes, with the "n" digit in the high-order nibble and the "n+1" digit in the low order. They are padded with the value #A (decimal 10).

cust_group_and_origtype

This two-byte type contains two pieces of information:

Cust_group_and_origtype digit organization

Byte	High-order nibble
Byte n	cust_grp_number (1)
Byte n+1	cust_grp_number (3)

The cust_grp_number is a three-digit value that represents the customer group of the originator, and the orig_type is the ORIG TYPE digit referred to in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119.

subgroup_and_termtype

The subgroup_and_termtype type provides information similar to the SUBGROUP and TERM TYPE fields described in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119. The format of these two nibble quantities is as follows:

Table 11
Subgroup_and_termtype digit organization

Byte	High-order nibble
Byte n	Subgroup number

route_info_and_date

The route_info_and_date field is made up of the two fields, ROUTE INFO DIGIT and DATE, described in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119. This information is formatted into the two bytes as follows:

Table 12
Route_info_and_date digit organization

Byte	High-order nibble
Byte n	Route Info Digit
Byte n+1	Date (2)

The three digits of the date describe the date as noted in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119. The Route Information Digit is described in the same document.

time_of_day

The time_of_day type follows the format of the TIME field specified in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119.

outpulsed_digits

The outpulsed_digits type contains digits in their normal fashion for all the bytes except the last. The last byte contains a digit indicating whether there were more than 29 digits outpulsed (overflow). The overflow value is "1" if there were more than 29 digits outpulsed, otherwise the value is "0".

Table 13
Outpulsed_digits digit organization

Byte	High-order nibble	Low-order nibble
Byte n	27th digit	28th digit
Byte n+1	29th digit	Overflow

record_type

The record_type value resides in the high order nibble of the byte. Its values and meaning are identical to the RECTYPE field described in the *Meridian Digital Centrex Station Message Detail Recording Reference Guide*, 297-2071-119.

Table 14
Record_type digit organization

Byte	High-order nibble	Low-order nibble
Byte n	Record Type	

Encoding send SMDR record RO

The encoding of the msl_send_smdr_call_record RO invocation follows a number of formats, depending on the SMDR call record format.

In previous paragraphs, the smdr_call_record was defined as a sequence of choice: the choice being an implicit octet string. This means that:

- The call data is a sequence of data elements.
- The individual data element identifiers have a class that is context-specific.
- The ID values 0-5 are associated with the different possibilities of the choice.
- The implicit type for the choice is an octet string.

This means that the other end expects an octet string for this context-specific class of information.

Therefore, for this sequence of call records, the data element identifiers are:

- #80 - D1 SMDR record (short)
- #81 - D2 Nerve record (short)
- #82 - D3 SMDR record (long)
- #83 - D4 Nerve record (long)
- #84 - D5 DAO extension
- #85 - D6 AUTH/ACCT extension
- #86 - D7 Not used
- #87 - D8 Not used
- #88 - D9 Not used
- #89 - DA ISUP extension
- #8A - DB NSS extension record

- #8B - DC TCN/PIN extension record
- #8C - Bearer capability record
- #8D - NCCI#7 billing ITC record
- #8E - SMDRITC feature ITC record

SMDR call record format examples

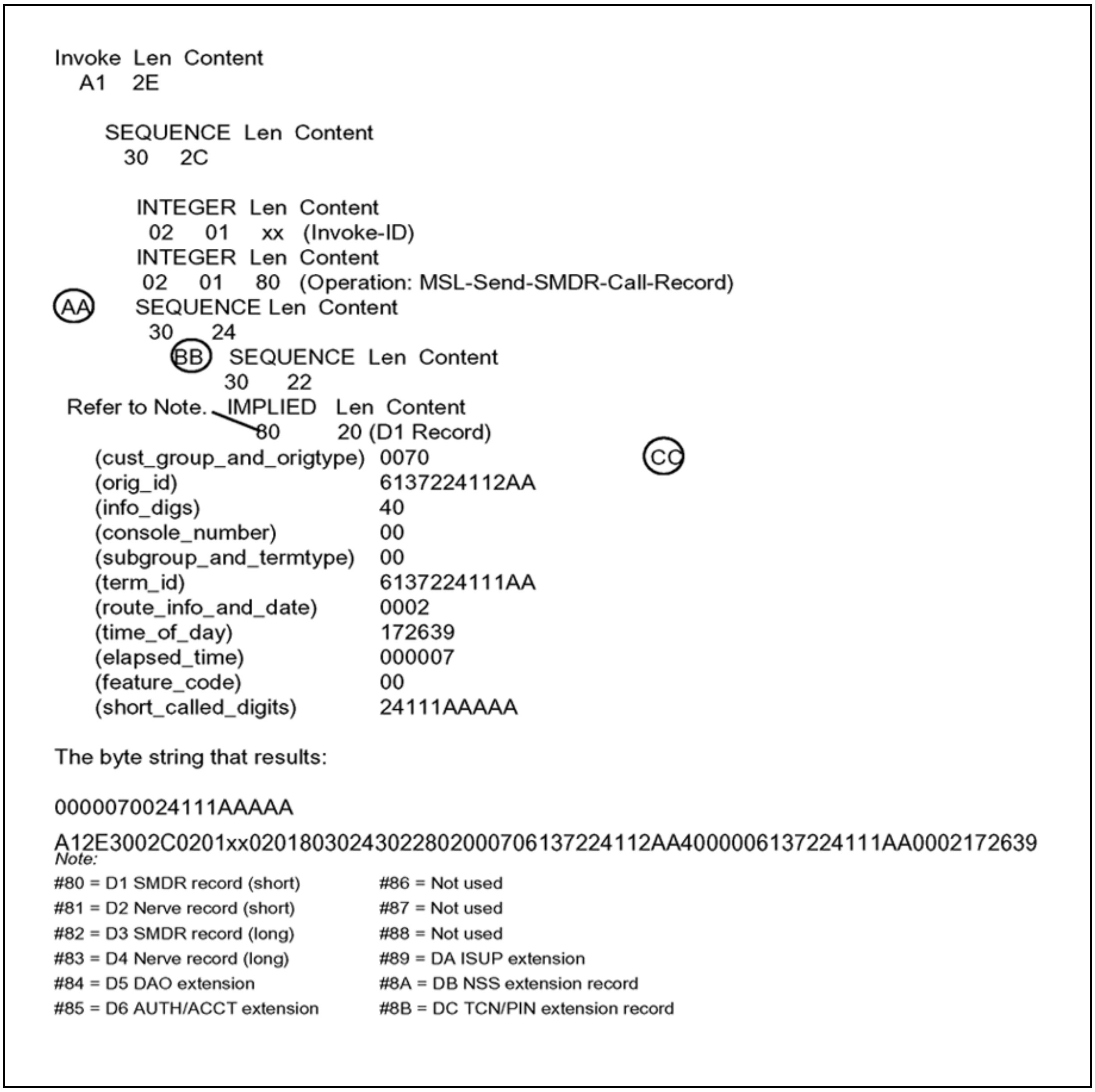
The example shown in Figure 5 "SMDR call record format" (page 30) is for a D1 format call record. The same basic rules apply to all of the record formats.

In Figure 5 "SMDR call record format" (page 30), the following comments apply:

- Line AA is a sequence that contains only the call record(s) data. This sequence can contain an extension-id as well as the call records (but does not in this case).
- Line BB is a sequence of call records. In this example there is only one call record in the sequence, but more are possible.
- Line CC is an implicit octet string that is the actual call data.

The example shown in Figure 6 "SMDR call record format with extension records (Part 1 of 2)" (page 31) and Figure 7 "SMDR call record format with extension records (Part 2 of 2)" (page 32) is for a D1 format record with two extension records needed to complete the call. This implies that an extension-id is contained in the RO. That extension-id is also present in an RO that contains the extension records.

Figure 5
SMDR call record format



```
Invoke  Len  Content
   A1    2E

      SEQUENCE  Len  Content
         30     2C

          INTEGER  Len  Content
            02     01    xx   (Invoke-ID)
          INTEGER  Len  Content
            02     01    80   (Operation: MSL-Send-SMDR-Call-Record)
 AA       SEQUENCE Len  Content
            30     24
             BB  SEQUENCE  Len  Content
                   30       22
 Refer to Note.  IMPLIED   Len  Content
                    80        20 (D1 Record)
     (cust_group_and_origtype)  0070                    CC
     (orig_id)                  6137224112AA
     (info_digs)                40
     (console_number)           00
     (subgroup_and_termtype)    00
     (term_id)                  6137224111AA
     (route_info_and_date)      0002
     (time_of_day)              172639
     (elapsed_time)             000007
     (feature_code)             00
     (short_called_digits)      24111AAAAA

The byte string that results:

0000070024111AAAAA

A12E3002C0201xx020180302430228020007061372241 12AA4000006137224111AA0002172639
```

Note:

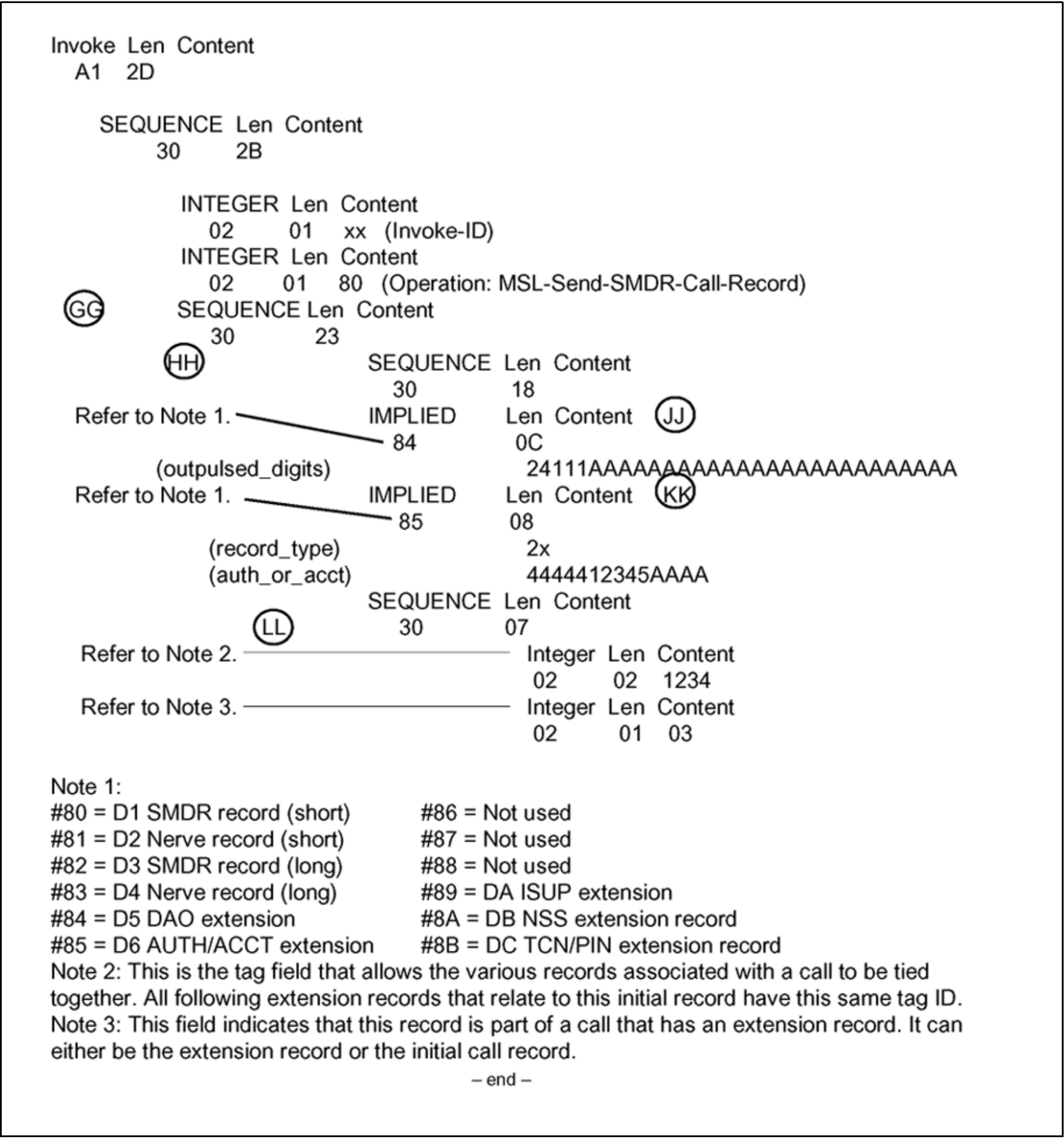
#80 = D1 SMDR record (short)	#86 = Not used
#81 = D2 Nerve record (short)	#87 = Not used
#82 = D3 SMDR record (long)	#88 = Not used
#83 = D4 Nerve record (long)	#89 = DA ISUP extension
#84 = D5 DAO extension	#8A = DB NSS extension record
#85 = D6 AUTH/ACCT extension	#8B = DC TCN/PIN extension record

Figure 6
SMDR call record format with extension records (Part 1 of 2)

```
Invoke  Len  Content
  A1    37

        SEQUENCE  Len  Content
          30      35

             INTEGER  Len  Content
               02     01   xx           (Invoke-ID)
             INTEGER  Len  Content
               02     01   80           (Operation: MSL-Send-SMDR-Call-Record)
   AA        SEQUENCE Len  Content
              30      2D
              BB           SEQUENCE  Len  Content
                             30      22
  Refer to Note 1. -->     IMPLIED   Len  Content
                             80      20
   (cust_group_and_origtype)  0070        CC
   (orig_id)                  6137224112AA
   (info_digs)                40
   (console_number)           00
   (subgroup_and_termtype)    00
   (term_id)                  6137224111AA
   (route_info_and_date)      0002
   (time_of_day)              172639
   (elapsed_time)             000007
   (feature_code)             00
   (short_called_digits)      24111AAAAA

            DD  SEQUENCE  Len  Content
                  30      07
  Refer to Note 2.           Integer  Len  Content   EE
                               02     02   1234
  Refer to Note 3.           Integer  Len  Content   FF
                               02     01   03
The byte string that results:
A13730350201xx020180302D3022802000706137224112AA4000006137224111AA000217
263900000700241 11AAAAA02021234020103
```

– continued –

Figure 7
SMDR call record format with extension records (Part 2 of 2)



```
Invoke  Len  Content
  A1    2D

     SEQUENCE  Len  Content
        30      2B

             INTEGER  Len  Content
               02      01    xx  (Invoke-ID)
             INTEGER  Len  Content
               02      01    80  (Operation: MSL-Send-SMDR-Call-Record)
  (GG)       SEQUENCE Len  Content
               30       23
        (HH)                  SEQUENCE  Len  Content
                                30       18
  Refer to Note 1. ------     IMPLIED    Len  Content  (JJ)
                                84        0C
         (outpulsed_digits)               24111AAAAAAAAAAAAAAAAAAAAAAAAAA
  Refer to Note 1. ------     IMPLIED    Len  Content  (KK)
                                85        08
            (record_type)                  2x
            (auth_or_acct)                 4444412345AAAA
                              SEQUENCE  Len  Content
               (LL)             30       07
  Refer to Note 2. ------------------------  Integer  Len  Content
                                              02      02   1234
  Refer to Note 3. ------------------------  Integer  Len  Content
                                              02      01   03
```

Note 1:

#80 = D1 SMDR record (short)	#86 = Not used
#81 = D2 Nerve record (short)	#87 = Not used
#82 = D3 SMDR record (long)	#88 = Not used
#83 = D4 Nerve record (long)	#89 = DA ISUP extension
#84 = D5 DAO extension	#8A = DB NSS extension record
#85 = D6 AUTH/ACCT extension	#8B = DC TCN/PIN extension record

Note 2: This is the tag field that allows the various records associated with a call to be tied together. All following extension records that relate to this initial record have this same tag ID.

Note 3: This field indicates that this record is part of a call that has an extension record. It can either be the extension record or the initial call record.

– end –

In Figure 6 "SMDR call record format with extension records (Part 1 of 2)" (page 31) and Figure 7 "SMDR call record format with extension records (Part 2 of 2)" (page 32), the following comments apply:

- Line AA is a sequence that contains some call information and an indicator that there is some extension information to be associated with the call.

- Line BB is a sequence of call records. In this example there is only one call record in the sequence, but more are possible.
- Line CC is an implicit octet string that is the actual call data.
- Line DD is a sequence containing the information necessary to identify extension records. It is generated for incoming calls provided the option Bearer Capability (BC) is datafilled in Table CUSTSMDR.
- The #DD SMDR extension record includes the following fields:
  - record code: value #DD. This is a two-character code that identifies the BC SMDR extension record.
  - bearer capability: BC. This field consists of a two-digit code that identifies the type of bearer capability. It holds the following bearer capability types based on originator:
    - nil (0)
    - speech (1)
    - 64K data (2)
    - 64K X.25 (3)
    - 56K data (4)
    - data unit (5)
    - 64K restricted (6)
    - 3.1Khz audio (7)
    - 7 kHz audio (8)
    - voice data (9)
  - bandwidth: This two-character field contains the bandwidth of the call. The valid range of this field is 1-24. The value "1" indicates a narrow band call. The values 2-24 represent the number of 64 K/b channels used in the call. The value stored in this field is in hexadecimal format.
- The #DD Extension Record is a billing record that is included in the following SMDR call record types:
  - D1 - short format SMDR record
  - D2 - short format NERVE CDR record
  - D3 - long format SMDR record
  - D4 - long format NERVE record
- Line EE is a two-byte integer value that is a unique identifier that appears on the associated extension record RO.

- Line FF is an integer value that indicates that there are two pieces of call information, indicated by the value "3", necessary to complete the call. These pieces of information can be contained in one or two associated ROs.

The example shown in Figure 6 "SMDR call record format with extension records (Part 1 of 2)" (page 31) and Figure 7 "SMDR call record format with extension records (Part 2 of 2)" (page 32) also contains two extension records that could be associated with the record for the preceding call. Both of the SMDR extension records are contained in the RO, along with an extension-id that specifies the SMDR call record with which these extension records are associated.

In Figure 6 "SMDR call record format with extension records (Part 1 of 2)" (page 31) and Figure 7 "SMDR call record format with extension records (Part 2 of 2)" (page 32), the following comments also apply:

- Line GG is a sequence that contains some extension information and an indicator that the extension information is to be associated with some other call record.
- Line HH is a sequence of extension records. There are two extension records in this sequence; it is possible to have just one.
- Line JJ is an implicit octet string that is one extension record (DAO format).
- Line KK is an implicit octet string that is the other extension record (AUTH-ACCT format).
- Line LL is a sequence containing the information necessary to identify the related call record.

MSL start transfer

The msl-start-transfer remote operation prepares the application on the receiving end to receive SMDR data. The RESULT and ERROR portions of the RO do not exist in the unidirectional link case. The transfer value indicates that SMDR data is to be transmitted. The transfer date and time is included for synchronization purposes. This RO is invoked whenever SMDR data is to be transferred across a link. See Figure 8 "Start transfer RO" (page 35).

The encoding of the msl-start-transfer RO invocation is as shown in Figure 9 "Encoding start transfer RO" (page 36).

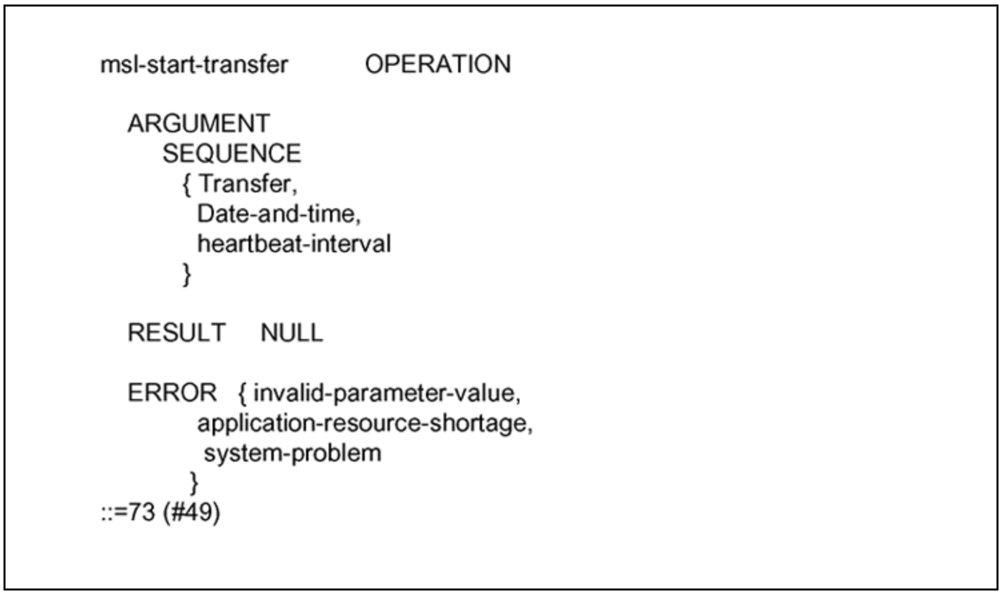
Transfer

The transfer definition is shown in Figure 10 "Transfer definition" (page 36). A value of "1" in this field indicates that SMDR call records are expected.

Date and time

The date_and_time type is defined as shown in Figure 11 "Date and time definition" (page 37).

Figure 8
Start transfer RO



```
msl-start-transfer        OPERATION

    ARGUMENT
        SEQUENCE
          { Transfer,
            Date-and-time,
            heartbeat-interval
          }

    RESULT    NULL

    ERROR   { invalid-parameter-value,
            application-resource-shortage,
             system-problem
           }
::=73 (#49)
```

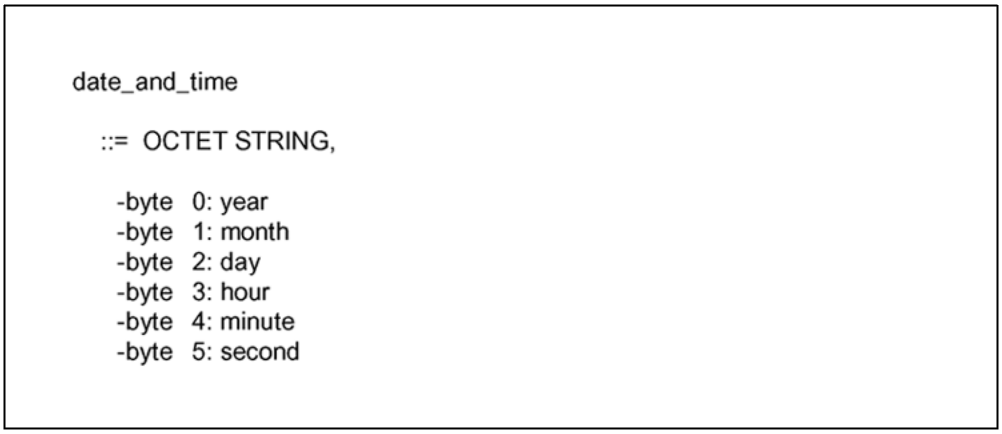
Figure 9
Encoding start transfer RO

```
Invoke  Len  Content
 A1     18

     SEQUENCE  Len  Content
       30     16

        INTEGER  Len  Content
         02      01   xx  (Invoke-ID)

        INTEGER  Len  Content
         02      01   49  (Operation: MSL-Start-xfr)

              SEQUENCE Len  Content
                30         0E

  Refer to Note 1.  ---------->INTEGER   Len  Content
                                02        01    01

  Refer to Note 2.  ---------->OCTET-   Len  Content
                               STRING
                                  04      06   56060A0B2112

  Refer to Note 3.  ---------->INTEGER  Len  Content
                                  02      01   0F

The byte string that results:
A11830160201xx020149300E020101040656060A0B211202010F

Note 1: This integer implies that the data to be transferred is SMDR data.
Note 2: An OCTET-STRING encoding of the date and time (for example,
06/10/86 at 11:33:18)
Note 3: Heartbeat duration value
```

Figure 10
Transfer definition

```
Transfer

     ::=  INTEGER
              { acd-mr-callevents (0)
                SMDR-call-record (1)
                }
```

Figure 11
Date and time definition



```
date_and_time

   ::=  OCTET STRING,

     -byte  0: year
     -byte  1: month
     -byte  2: day
     -byte  3: hour
     -byte  4: minute
     -byte  5: second
```

MSL stop transfer

The msl-stop-transfer remote operation terminates transmission of data on a link currently accepting data. See Figure 12 "Stop transfer RO" (page 38). The data stream that is being halted is defined by the type Transfer, which is shown in the msl-start-transfer RO, Figure 8 "Start transfer RO" (page 35).

The encoding of the msl-stop-transfer RO invocation is as described in Figure 13 "Encoding stop transfer RO" (page 38).

MSL connect and disconnect

The connect and disconnect RO defined in Figure 15 "Connect and disconnect RO" (page 39) are also defined for the ACD management reporting system. These two operations initialize the link for communication and establish the protocol version to be used for the link.

Figure 12
Stop transfer RO

```
msl-stop-transfer        OPERATION

  ARGUMENT
          SEQUENCE
              {
                Transfer,
              }

  RESULT          NULL

  ERROR        {
                 operation-sequence-problem
               }
::=7 6      (#4C)
```

Figure 13
Encoding stop transfer RO

```
Invoke  Len  Content
  A1    0D

        SEQUENCE  Len  Content
           30          0B

                INTEGER  Len  Content
                  02       01   xx  (Invoke-ID)
                INTEGER  Len  Content
                  02       01   4C  (Operation: MSL-Stop-xfr)
                SEQUENCE Len  Content
                   30      03

Refer to Note———>INTEGER   Len  Content
                    02       01   01

The byte string that results:
A10D300B0201xx02014C3003020101

Note: This INTEGER value implies that the SMDR data stream is stopping.
```

The only information type used for this pair of operations is the protocol version, which is the string defined to be the BCS name. It is defined in Figure 14 "Protocol version" (page 39).

Figure 14
Protocol version

```
protocol_version :: = IA5STRING
```

An example of this might be the string "BCS21". The string always contains six characters. The space at the end of the string in the preceding example is considered a character.

Figure 15
Connect and disconnect RO

```
msl-connect              OPERATION

    ARGUMENT
              SEQUENCE
              { Protocol-version
                }

   RESULT
             SEQUENCE
                  { Protocol-version
                    }

   ERROR          { invalid-parameter-value,
                    application-resource-shortage,
                  operation-sequence-problem,
                    system-problem
                    }
::=64     (#40)


msl-disconnect              OPERATION

      ARGUMENT    NULL

       RESULT      NULL

   ERROR          { operation-sequence-problem }

::=65     (#41)
```

Allocation for inbound toll calls

The SMDR Allocation for Inbound Toll Calls feature enables SMDRs for incoming calls on a per-trunk-group basis. This feature provides a separate record of each party involved in the call from answer to disconnect. The feature objective is to split the cost of an incoming toll call among the destination numbers to which the call connects. The detection of an inbound toll call on an incoming or two-way trunk that has the SMDR Incoming Toll Call (SMDRITC) option datafilled in Table TRKGRP activates this feature.

SMDR extension record

When a call is answered and then transferred on a trunk group having this feature, the feature creates an extension record as part of the trunk group SMDR record. Each SMDRITC extension record has two fields, a CALLID and a time stamp. The CALLID is the same for every SMDRITC extension record generated for a particular call. The time stamp represents the time at which billing for this SMDRITC extension record stops and billing for the next SMDRITC record begins.

SMDR extension record billing

If a particular trunk group has the SMDR feature, the call is billed by means of the extension record detailing how long the call is present at each terminating station.

Billing the terminating destination number begins only when a station answers. Billing continues for the time the call is held at that station plus the time to either answer at another station in the case of call transfer or to disconnect. If a destination number transfers a call to another destination number and that station does not answer the call for several seconds, the originating station receives the bill for those intervening seconds even if the originating station disconnects prior to the second destination number answering the call. The SMDR record extension exists only if the call is an inbound toll call on a trunk group with the SMDRITC feature activated. This state designates the call as billable and allows SL-100 system users to allocate the cost of an inbound toll call to the appropriate departments that took part in the call.

Billing interactions for SMDRICT

Automatic Message Accounting (AMA) bills three-way calling to the controller of the call. It generates a call record and bills the controller for the length of time the controller is in the call. When the controller of the call disconnects, billing and a new call record start on the called party designated to take control of the call.

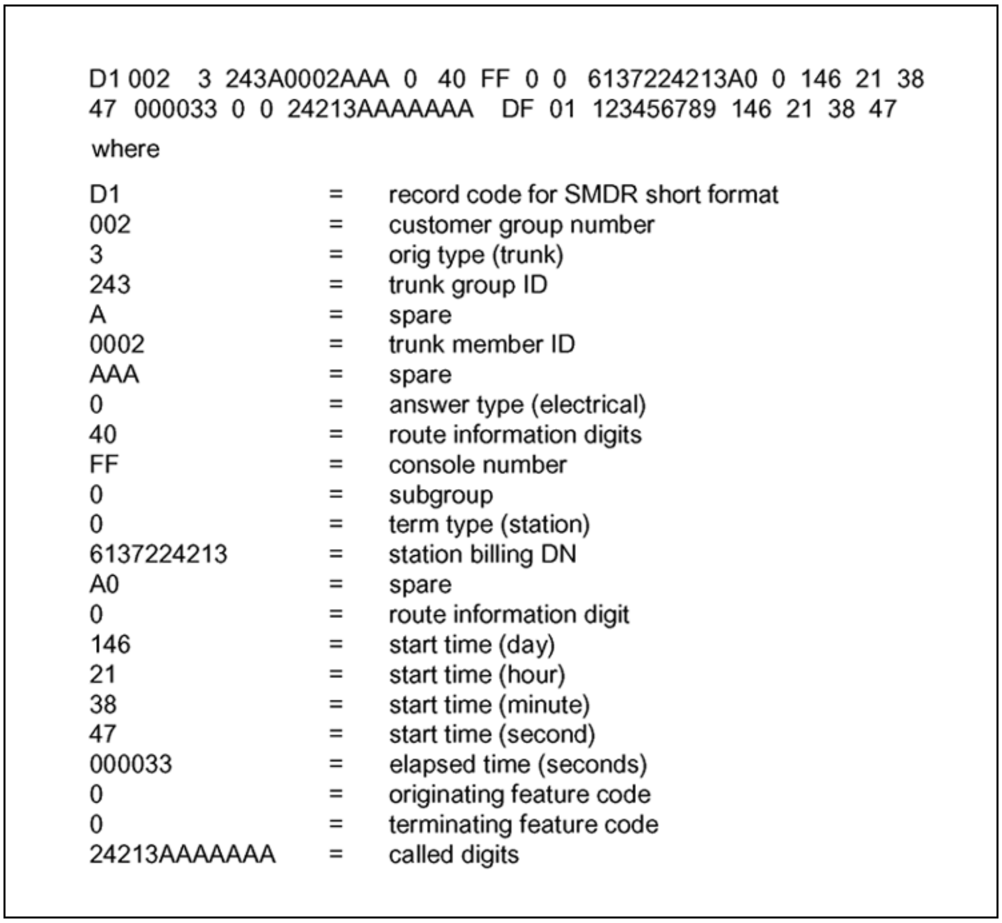
Restrictions for SMDRITC

SMDRITC is a valid option for IBNT1 and IBNT2 trunk group types in Table TRKGRP. It only applies to calls designated as inbound or two-way toll calls over a trunk group.

Example of SMDR containing the inbound toll extension record

The following example depicts an SMDR D1 record that contains the generic SMDR extension record code, DF, as well as the Inbound Toll Call (ITC) extension record, 01:

Figure 16
SMDR D1 record

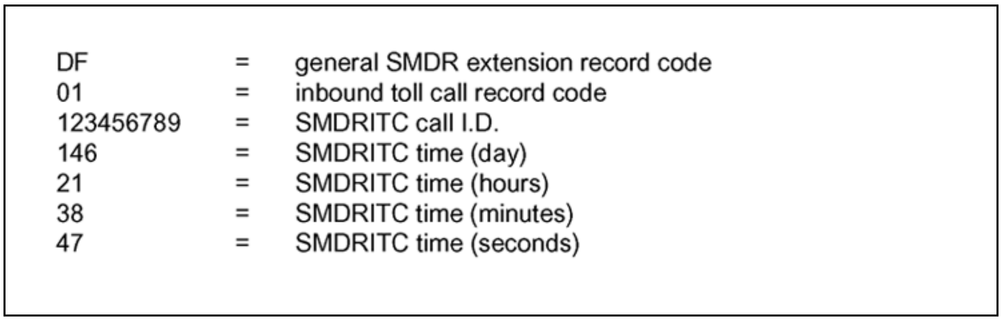


```
D1 002   3  243A0002AAA  0   40  FF  0  0   6137224213A0  0  146  21  38
47  000033  0  0  24213AAAAAAA   DF  01  123456789  146  21  38  47
```

where

D1	=	record code for SMDR short format
002	=	customer group number
3	=	orig type (trunk)
243	=	trunk group ID
A	=	spare
0002	=	trunk member ID
AAA	=	spare
0	=	answer type (electrical)
40	=	route information digits
FF	=	console number
0	=	subgroup
0	=	term type (station)
6137224213	=	station billing DN
A0	=	spare
0	=	route information digit
146	=	start time (day)
21	=	start time (hour)
38	=	start time (minute)
47	=	start time (second)
000033	=	elapsed time (seconds)
0	=	originating feature code
0	=	terminating feature code
24213AAAAAAA	=	called digits

Figure 17
Inbound toll call extension record fields



DF	=	general SMDR extension record code
01	=	inbound toll call record code
123456789	=	SMDRITC call I.D.
146	=	SMDRITC time (day)
21	=	SMDRITC time (hours)
38	=	SMDRITC time (minutes)
47	=	SMDRITC time (seconds)

Office datafill for SMDRITC option

The user datafills the inbound extension record on a per-trunk-group basis in Table TRKGRP. The SMDRITC option applies to incoming and two-way trunks only.

SMDRITC data destination designation

The user selects the destination of the data by datafill and by using the Device Independent Recording Package (DIRP).

Generic extension code

The formatter utility uses the record code to determine what information needs to be output for a particular SMDR record. "DF" designates generic extension record code. The first field in the "DF" extension record, following the record code field, is the format code. The format code field defines the format of the remainder of the "DF" extension record. The extension record for SMDR inbound toll call uses the format code "01".

SMDRITC related features

The SMDRITC feature needs the Three-Way Call and the Call Transfer features to function properly.

Protocol sequences

The RO defined in the previous portions of this publication have meaning in certain sequences. The appropriate sequences concern:

- connecting the link
- establishing that SMDR data is to be passed across the link
- sending SMDR data records

In the case of extension records being used for SMDR records, the main record (#D1-#D4) is transmitted, and then the extension records (#D5, #D6) are transmitted. The use of an extension-id allows extension records to be correlated with the main call record.

Table Table 15 "Data link states and sequences" (page 43)3 shows the datalink states and describes the various conditions and RO sequence. Any RO invoked out of sequence is ignored.

After all restart conditions, the link disconnects and returns to the idle state.

Valid data link states

Table 15 "Data link states and sequences" (page 43) Table 15 "Data link states and sequences" (page 43)describes all of the valid transitions as a result of various events. The valid states are:

- idle - inactive, passing no information. This is the state achieved after all restarts or after all link failure conditions.
- connected - establishes interapplication for either SMDR or ACD management report applications
- SMDRing - indicates passing SMDR information

Table 15
Data link states and sequences

Current state	Event	Next state	Remarks
Idle	MSL- Connect	Connected	This is the only valid transition from this state.
Connected	MSL- Start- (SMDR) XFR	SMDRing	Another valid message at this state starts up ACD management reporting.
Connected	<Error>	Idle	Any restart or severe link failure causes this transition.
Connected	MSL- Disc.	Idle	
Connected	<any RO>	Connected	Anything false is ignored.
SMDRing	MSL- Stop- (SMDR) XFR	Connected	This is the only non-error transition from this state.
SMDRing	<any RO>	SMDRing	Anything false is ignored.
SMDRing	<Error>	Idle	Error, failure, and restart conditions cause a return to the idle state.

Feature description

Data link characteristics

The data link used for the Station Message Detail Recording (SDMR) Data Access feature consists of an RS-232-C interface used to transmit ASCII characters. Data flow on the link is two-way between the Meridian SL-100 system and the Down Stream Processor (DSP). No flow control, error detection, error correction, or end-to-end protocol is provided. The transmission speed of the data link is limited by the capacity of the NT1X67 terminal controller card.

A maximum of sixteen data links are supported by each Meridian SL-100 system. A data link device must be datafilled in Table TERMDEV before it can be referenced by the management reports Command Interpreter (CI) interface. A data link device must also be datafilled in Table SLLNKDEV before it can be connected in the LNKUTIL CI increment. This device must be datafilled in Table TERMDEV before it can be datafilled in Table SLLNKDEV.

Successful SMDR data flow for this feature depends on the links' status and maintenance provided by the CI increment, LNKUTIL. If the LNKUTIL function is absent from the load, then the linkage between the Meridian SL-100 system and the DSP is undefined.

A set of CI commands allows basic maintenance and manipulation of the data links. The user has the capability to perform the following tasks:

- start and stop all SMDR data flow on a specific data link
- add data links to the system
- delete data links from the system
- query data link status
- initialize transmission across the data link
- prevent transmission across the data link

Refer to "Human-machine interface" (page 51) for LNKUTIL and SMDRLNK commands.

Setting up the Terminal Server

This section covers the commands that should be entered to set-up the terminal server. By setting up a terminal server, you are configuring a dedicated telnet service on an in reach access server.

> ***Note:*** Certain settings, such as the speed and flow control, must match the settings that have been configured on the attached serial device. The following commands give an example using speed of 9600, 8-none-1, and XON/software flow control. If your serial device happens to be configured for 7-even-1, or a different speed, or uses CTS/hardware flow control, then be sure to use those settings when issuing the following commands.

Enter the following commands in Privilege Mode.

```
iTouch_Priv> define port # telnet dedicated [ip_addr]:[telnet
_remote#]

iTouch_Priv> define port # speed 9600

iTouch_Priv> define port # flow control XON (default setting)

iTouch_Priv> define port # character size 8 (default setting)

iTouch_Priv> define port # parity none (default setting)

iTouch_Priv> define port # stop bits 1 (default setting)

iTouch_Priv> define port # internet connections enabled
```

Additional Notes and Settings

If the telnet remote port number is 23, you do not have to specify it in the command line.

If you are making a dedicated telnet session from one access server to another access server, then the following configuration is required. On the port of the remote server to which the connection is being made, enter the following commands.

```
iTouch_Priv> define port # outboundsecurity disabled

iTouch_Priv> define port # internet connections enabled
```

If you want this dedicated session to be permanently established, you will need to enable "autodedicated" on the local server port. This will cause the port to try to re-establish the connection if it is ever dropped or if the user logs out of the host. You must also have "autobaud’ disabled in order for this feature to work. Enter these commands to permanently establish a dedicated session.

```
iTouch_Priv> define port # autodedicated enabled

iTouch_Priv> define port # autobaud disabled
iTouch_Priv> define port # speed 9600
```

> ***Note:*** The speed entered must match the speed set on the attached serial device.

Defining data links

SMDR data links are defined by the SMDRLNK CI increment using the Maintenance and Administrative Position (MAP) workstation. The SMDRLNK CI increment contains the basic commands that are unique to SMDR reports. The following commands are included:

- SMDRLNK - this command enters the SMDRLNK CI increment. It initializes the directory, adds the basic command set to the directory, extends the symbol table, and adds the directory to the symbol table. The SMDRLNK can only be exited through use of the QUIT command. If the user tries to enter this level more than once, a general information message about the SMDRLNK increment displays.
- QUIT - this command enables a craftsperson to leave the SMDRLNK CI increment. It removes the directory from the symbol table, shrinks the symbol table, and deallocates the directory.
- SENDSMDR - this command routes SMDR call records to a data link pool. If the pool does not exist, an informative message generates, and no further action is taken. If the pool exists and contains more than one data link, then SMDR records are randomly distributed among data links.
- STOPSMDR - this command deletes routing information for SMDR call records to a specified data link pool. The pool is still known to the system, but SMDR transfer has been removed for that pool. If no routing information had been specified previously, then no action is taken.
- SMDRSTAT - this command queries routing information about SMDR call records. Depending upon the query of one or all pools, the states of the pool and their devices display.

The LNKUTIL CI command, DEVCON, defines a device (data link) as a member of a specific pool. Pools are logical entities that can be specified for customer groups, functions, type of reports, or other groupings. Data links can also be assigned or reassigned to particular pools for load-sharing reasons. A device can be defined as a member of more than one pool.

Refer to "Human-machine interface" (page 51) for further usage information for these commands.

Logs

Logs provide a hard copy history of activities on each data link. The logs record the information on start and stop of data transfer, start and stop of call event message generation, and error conditions. Refer to "Log reports" (page 65) for additional information on SMDR-related logs.

Table SLLNKDEV

Table SLLNKDEV specifies characteristics of data links used by the LNKUTIL CI increment. It enables the device-connecting procedure to make use of the characteristics of the device.

All devices must be datafilled in Table SLLNKDEV before they are connected in the LNKUTIL CI increment. These devices must be datafilled in Table TERMDEV before they can be datafilled in Table SLLNKDEV. Refer to the *Customer Data Schema,* 555-4031-851, or the *Defense Switched Network Customer Data Schema*, 555-402-851.

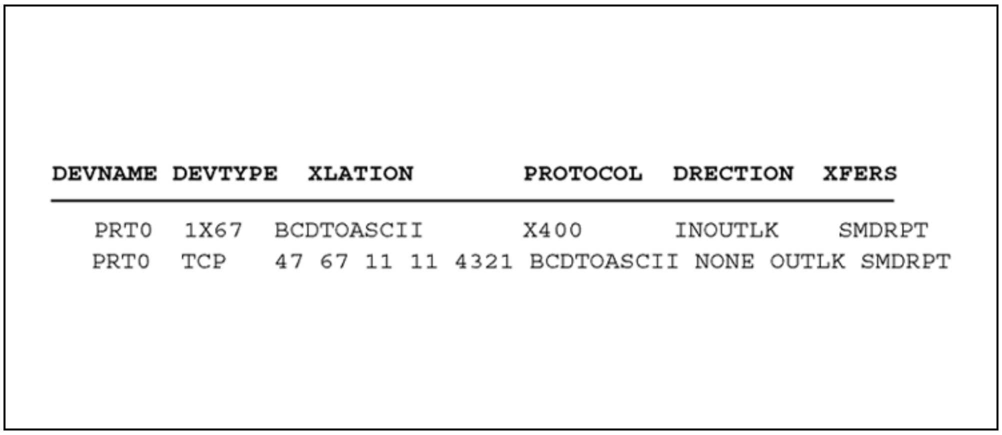
Table Table 16 "Table SLLNKDEV datafill" (page 48) describes the fields to be entered. Figure 18 "Specifying a data link device using Table SLLNKDEV" (page 50) shows example datafill for Table SLLNKDEV.

Table 16
Table SLLNKDEV datafill

Field	Subfield	Explanation	Action
DEVNAME		Device name	Enter the 1- to 16-character device name to be used in LNKUTIL.
DEVTYPE	DEVICE	Device type	This field consists of the subfield DEVICE. Enter TCP, IP address, and port number. For example, "TCP 47 67 11 11 4321". Enter "1X67" to specify the link on IOC or IOM.
XLATION		Translation	Enter "NONE" or "BCDTOASCII" to specify the SL-100-to-link translation used for outgoing and incoming data links.
PROTOCOL		Protocol	Enter "NONE" or "X400" to specify the protocol that is expected by the data link and the SL-100 system concerning the connection and starting messages, as well as any leading byte information required.

Field	Subfield	Explanation	Action
DRECTION		Direction	Enter one of the following values to indicate the direction in which the data travels through the data link: • INLK = In-link • OUTLK = Out-link • INOUTLK = In/out-link
CONTMARK		Continuation mark	Enter "+" to indicate that additional information for this tuple is contained in the next record.
XFERS		Transfers	Enter one of the following values to specify the allowable report types on the data link: • ACDRTD = ACD Real Time Display • MGTRPT = Management Reports • SMDIDATA = SMDI Data • SMDRRPT = SMDR Reports • XSMDATA = XSM Data If ACDRTD or SMDIDATA reports are entered, no other types are allowed on this data link. MGTRPT and SMDRRPT reports can be specified on the same data link. This field accepts a vector with up to five entries. Enter "$" to terminate the vector.
OPTION		Option	Refer to the *Customer Data Schema*, 555-4031-851 or 555-4021-851, for a detailed list of available options.

Figure 18
Specifying a data link device using Table SLLNKDEV



```
DEVNAME DEVTYPE  XLATION        PROTOCOL  DRECTION  XFERS
_________________________________________________________
   PRT0   1X67   BCDTOASCII       X400      INOUTLK    SMDRPT
   PRT0   TCP    47 67 11 11 4321 BCDTOASCII NONE OUTLK SMDRPT
```

Human-machine interface

The management reports application layer provides a generic data link interface to transfer Station Message Detail Recording (SMDR) call records data to a Down Stream Processor (DSP). A set of Command Interpreter (CI) commands can be used to manipulate the data links.

A CI increment called LNKUTIL maintains and controls data links used for data transfer between the Meridian SL-100 system and any other machine that can support the necessary protocol.

A CI increment level called SMDRLNK, which utilizes LNKUTIL, is created by this feature.

The LNKUTIL CI increment provides sequence control, routing information, and status information functions for Automatic Call Distribution (ACD) management reports in the ACDPOOLS CI increment. This feature enhances the LNKUTIL level to include information functions about SMDR reports and adds the SMDRLNK CI increment, which includes functions unique to SMDR reports.

All data links must be datafilled in Table SLLNKDEV before they can be connected in the LNKUTIL CI increment.

Using the MAP workstation

The SMDR Data Access commands are performed using Human-Machine Interface (HMI). The HMI consists of the Maintenance and Administration Position (MAP) workstation and the software required to convert human information to machine information and vice versa.

The hardware and software conditions of the system, as well as maintenance action and responses, display at the MAP workstation. Information is presented in the form of a basic display.

The character string, "CI:", and the character, ">", display on two lines in the upper left-hand corner of the MAP screen. They indicate that the MAP workstation is ready to interact with the Command Interpreter in analyzing data. This basic display operates in the scroll mode.

The scroll mode advances each entered line to the next one higher each time an additional line is entered. When all available lines on the display have been used and additional lines are entered, the uppermost line scrolls to the top of the screen and disappears.

LNKUTIL commands

LNKUTIL

The LNKUTIL command causes the system to respond by placing the user in LNKUTIL CI increment. The prompt `LNKUTIL`: displays. The user stays in the LNKUTIL increment and has access to LNKUTIL commands. There are no parameters, range, or defaults associated with the LNKUTIL command.

In LNKUTIL, the following commands are available to the user:

- DEVCON
- DEVDISC
- DEVSTART
- DEVSTOP
- LNKSTAT
- POOLSTART
- POOLSTOP
- QUIT

The user can obtain help on the format of any of these commands by entering the following command:

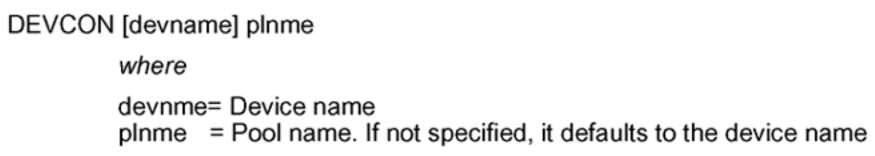
```
Q <commandname>
```

DEVCON

The DEVCON command enables a transfer session on a specific data link. It can also define a device as a member of a pool. If a pool name is not specified, it defaults to the device name. A log report, SLNK100, generates whenever the DEVCON command is entered and the device link is connected from a previously disconnected state.

Command format

The format of the DEVCON command is as follows:



```
DEVCON [devname] plnme
          where
          devnme= Device name
          plnme   = Pool name. If not specified, it defaults to the device name
```

Examples of the DEVCON command

The following are examples of DEVCON command syntax:

```
DEVCON PRT0 FIRST
```

```
DEVCON SECOND
```

System responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If no problems are encountered, the response is:

  ```
  Device PRT0 has been started.
  ```

- If the datalink is not datafilled in table SLLNKDEV, the response is:

  ```
  Specified datalink is not datafilled in table SLLNKDEV.
  No action taken.
  ```

- If the maximum number of links has been assigned to the specified pool, the response is:

  ```
  The number of datalinks assigned to the pool FIRST is 4.
  No more datalinks may be assigned.  No action taken.
  ```

- If the maximum number of data links has been reached, the response is:

  ```
  Unable to allocate device PRT0.
  ```

- If the maximum number of pools has been reached, the response is:

  ```
  Unable to allocate pool FIRST.
  ```

- If the current device status is not DISCONNECTED or DEAD, the response is:

  ```
  Device PRT0 has already been started.
  ```

- If the system is unable to send a Meridian SL-100 (MSL) connect Remote Operation to the DSP, the response is:

  ```
  Unable to start device PRT0.  No action taken.
  ```

DEVDISC

The DEVDISC command disables a transfer session on the specified data link or deletes information about the data link from the system. If the DEVDISC command is entered without the KILL parameter, an MSL-disconnect remote operation goes to the DSP, and the Meridian SL-100 file system interface for the data link is deallocated. If the KILL parameter is specified, the same two actions are performed: the device information is deleted, and the device is removed from its assigned pool. If the device is the only device assigned to the pool, and no other applications are referencing the pool, the pool information is deleted as well. A log report, SLNK101, generates whenever the DEVDISC command is entered, and the device link is disconnected from a previously-connected state.

Command format

The format of the DEVDISC command is as follows:

```
DEVDISC[devnme]  KILL
          where
          devnme= Device name
```

Examples of the DEVDISC command

The following are examples of DEVDISC command syntax:

```
DEVDISC PRT0

DEVDISC PRT0 KILL
```

System responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If no problems are encountered, the response is:

  ```
  Device PRT0 has been stopped.
  ```

- If the KILL parameter is entered, the response is:

  ```
  Device PRT0 has been stopped.  Device PRT0 has been
  deleted from pool FIRST.
  ```

- If the current device status is DISCONNECTED or DEAD, the response is:

  ```
  Device PRT0 is not in a connected state.  No action taken.
  ```

- If the system is unable to send a MSL-disconnect Remote Operation to the Down Stream Processor, the response is:

  ```
  Unable to stop device PRT0.
  ```

DEVSTART

The DEVSTART command starts data transfer for the specified data link. The device must be started before the transfer can be initiated. As long as the link status is CONNECTED, any system objections to the start of data transfer can be overruled by using the FORCE option. A log report, SLNK102, generates whenever the DEVSTART command is entered and is valid.

Command format

The format of the DEVSTART command is as follows:

```
DEVSTART devnme SMDRRPT [FORCE]
            where
            devnme= Device name
```

Examples of the DEVSTART command

The following examples show DEVSTART command syntax:

```
DEVSTART PRT7 SMDRRPT

DEVSTART PRT7 SMDRRPT FORCE
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If no problems are encountered, the response is:

  ```
  SMDRRPT transfer has been started on device PRT7.
  ```

- If the device has not been datafilled in Table SLLNKDEV for SMDR Reports, the response is:

  ```
  SMDRRPT has not been datafilled in table SLLNKDEV for
  PRT7.It may not be used for SMDR Reports.  No action
  taken.
  ```

- If the current device status is DISCONNECTED or DEAD, the response is:

  ```
  Device PRT7 has not been started.  No action taken.
  ```

- If the system is unable to send a MSL-start-transfer Remote Operation to the Down Stream Processor, the response is:

  ```
  Unable to start SMDRRPT transfer on device PRT7.No action
  taken
  ```

DEVSTOP

The DEVSTOP command stops data transfer for the specified datalink. The device must be known to the system. Transfer must be started before it can be stopped. A log report, SLNK103, generates whenever the DEVSTOP command is entered and is valid.

Command format

The format of the DEVSTOP command is as follows:

```
DEVSTOP devnme SMDRRPT
        where
        devnme= Device name
```

Example of the DEVSTOP command

The following is an example of DEVSTOP command syntax:

```
DEVSTOP PRT7 SMDRRPT
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If no problems are encountered, the response is:

  ```
  SMDRRPT transfer has been stopped on device PRT7.
  ```

- If the current device status is not TRANSFERRING, the response is:

  ```
  SMDRRPT transfer has not been started on device PRT7.No
  action taken.
  ```

- If the system is unable to send a MSL-stop-transfer Remote Operation to the Down Stream Processor, the response is:

  ```
  Unable to stop SMDRRPT transfer on device PRT7.
  ```

LNKSTAT

The LNKSTAT command displays information on all the data links in the Meridian SL-100 Link (SLLNK) Operational Measurement (OM) system. The device or pool must be known to the system. This command does not affect data link status.

Command format

The format of the LNKSTAT command is as follows:

```
LNKSTAT  DEVICE devnme
         POOL plnme
         ALL
         where
         devnme= Device name
```

Examples of the LNKSTAT command

The following are examples of LNKSTAT command format:

```
LNKSTAT DEVICE MAP

LNKSTAT POOL FIRST

LNKSTAT ALL
```

System Responses

The following is a sample response for the LNKSTAT DEVICE MAP command:

```
POOL DEVICE STATUS DATA STREAM
FIRST MAP Transferring SMDR Reports
```

The following is a sample response for the LNKSTAT POOL FIRST command:

```
POOL DEVICE STATUS DATA STREAM
FIRST MAP Disconnected
```

The following is a sample response for the LNKSTAT ALL command:

```
FIRSTMAPADisconnected
SECONDMAPBTransferring SMDR Reports
THIRDMAPCConnected
FOURTHMAPDTransferring ACD Management
Reports
Transferring SMDR Reports
```

POOLSTART

The POOLSTART command starts data transfer on all devices in the specified pool for the specified transfer type. It is equivalent to doing a DEVSTART on each device in the pool for the report type specified. If any device is not in a state that allows data transfer, no data transfer occurs. A log report, SLNK102, generates whenever the POOLSTART command is entered and is valid.

Command format

The format of the POOLSTART command is as follows:

```
POOLSTART  plnme  SMDRRPT
           where
           plnme   = Pool name
```

Example of the POOLSTART command

The following example shows POOLSTART command syntax:

```
POOLSTART COLLECT SMDRRPT
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure is displays.

- If there is an allowable number of links in pool COLLECT, the following response is given:

  ```
  SMDRRPT transfer has been started on device PRT1.
  ```

- If one or more links in the pool is currently transferring SMDRRPT, the following response is given:

  ```
  Transfer on PRT1 in pool COLLECT has already been started.
  Transferring on pool COLLECT may not be started.  No
  action taken.
  ```

- If device PRT7 in pool COLLECT has not been datafilled, the response is:

  ```
  SMDRRPT is not datafilled in table SLLNKDEV for PRT7.  It
  may not be used for SMDR Reports.  No action taken.
  ```

POOLSTOP

The POOLSTOP command stops data transfer on all devices in the specified pool for the specified transfer type. It is equivalent to doing a DEVSTOP on each device in the pool for the report type specified. If any device is not transferring, nothing happens. A log report, SLNK103, generates whenever the POOLSTOP command is entered and is valid.

Command format

The format of the POOLSTOP command is as follows:

```
POOLSTOP   plnme  SMDRRPT
           where
           plnme   = Pool name
```

Example of the POOLSTOP command

The following is an example of POOLSTOP command syntax:

```
POOLSTOP COLLECT SMDRRPT
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If all of the data links are currently transferring SMDRRPT, the following response is given:

  ```
  SMDRRPT transfer has been stopped on device PRT0.SMDRRPT
  transfer has been stopped on device PRT1.
  ```

- If one or more links in the pool are not currently transferring SMDRRPT, the following response is given:

  ```
  SMDRRPT transfer has not been started on device
  PRT1.Transferring on pool COLLECT may not be stopped.  No
  action taken.
  ```

QUIT

The QUIT command causes the system to respond by leaving the LNKUTIL CI increment. The user no longer has access to LNKUTIL commands. There are no parameters, range, or defaults associated with the QUIT command.

SMDRLNK commands

SMDRLNK

The SMDRLNK command causes the system to respond by placing the user in SMDRLNK CI increment. The prompt SMDRLNK: displays. The user stays in the SMDRLNK increment and has access to SMDRLNK commands.

The SMDRLNK CI increment is entered and exited through the LNKUTIL CI level. In SMDRLNK, the following commands are available to the user:

- QUIT
- SENDSMDR
- SMDRLNK
- SMDRSTAT
- STOPSMDR

The user can obtain help on the format of any of these commands by entering the following command:

```
Q <commandname>
```

QUIT

The QUIT command causes the system to respond by leaving the SMDRLNK CI increment. The user no longer has access to SMDRLNK commands. There are no parameters, range, or defaults associated with the QUIT command.

SENDSMDR

The SENDSMDR command routes SMDR information to a specified pool for the subsequent call analysis. At least one pool and one device must be known to the system (by the DEVCON command) for the SENDSMDR command to work.

Command format

The format of the SENDSMDR command is as follows:

```
SENDSMDR   plnme
           where
           plnme   = Pool name
```

Example of the SENDSMDR command

The following is an example of SENDSMDR command syntax:

```
SENDSMDR FIRST
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

If **SENDSMDR** is entered, the device is known to the system, and the DEVSTART command has been given, the following response is given:

```
SMDR reports have been routed to pool FIRST.
```

If **SENDSMDR FIRST** is entered, but the pool is not known to the system, the following response is given:

```
Pool FIRST does not exist.  No action taken.
```

STOPSMDR

The STOPSMDR command disables transmission of SMDR data that was previously assigned to a specific pool. It requires a pool parameter to identify which pool to stop. This command overrides the DEVSTOP command and causes all devices assigned to the pool specified to stop transferring SMDR reports. No action is taken if the SENDSMDR command has not been given.

Command format

The format of the STOPSMDR command is as follows:

```
STOPSMDR   plnme
           where
           plnme   = Pool name
```

Example of the STOPSMDR command

The following is an example of the STOPSMDR command:

```
STOPSMDR FIRST
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If no problems are encountered and SMDR is routed to FIRST, the following response is given:

  ```
  SMDR reports for pool FIRST have been stopped.
  ```

- If no SMDR reports have been previously routed to any pool, the following response is given:

```
SMDR Reports have not been assigned to any pool.  No
action taken.
```

- If SMDR reports are not routed to any specified pool, the following response is given:

  ```
  SMDR Reports have not been assigned to pool FIRST
  ```

SMDRSTAT

The SMDRSTAT command displays data transfer information about SMDR reports and related data links. This command does not affect data link status.

Command format

The format of the SMDRSTAT command is as follows:

```
SMDRSTAT  POOL   plnme
          ALL
```

where

plnme	= Pool name

Examples of the SMDRSTAT command

The following is an example of the SMDRSTAT command:

```
SMDRSTAT POOL SECOND

SMDRSTAT ALL
```

System Responses

A message displays to indicate the success or failure of the request. If the request failed, the reason for failure displays.

- If SMDRSTAT POOL SECOND is entered and a SENDSMDR command operation was previously done, the following response is given:

  ```
  SMDR reports have been routed to pool SECOND.
  ```

- If SMDRSTAT POOL SECOND is entered and no previous SENDSMDR command operation was done, the following response is given:

  ```
  No SMDR reports have been routed to pool SECOND.
  ```

- If SMDRSTAT ALL is entered and several SENDSMDR commands were done successfully to pool TOP and BOTTOM, the following response is given:

  ```
  SMDR Reports have been routed to pool BOTTOM
  with the following devices:
  MAP1, MAP2
  SMDR Reports have been routed to pool TOP with
  ```

```
the following devices:
MAP5
```

- If SMDRSTAT ALL is entered and no previous SENDSMDR command was successfully done, the following response is given:

```
No SMDR reports have been routed.
```

Operational measurements

Operational measurement group SLLNK

The Operational Measurement (OM) group SLLNK aids in reporting the link status, which includes the number of messages it can and cannot handle. It provides the following measurements for the data link utilities pertaining to SMDR reports transfer messages:

- SLLNKOVF - the number of messages that are overwritten or discarded due to a full queue
- SLLNKOK - the number of messages enqueued successfully to be transferred across the datalinks
- SLLNKQU - the number of messages in the queue waiting to be processed. This figure updates every 100 seconds. Averaging is done by dividing this number by the number of times slow samples were taken.

Station Message Detail Recording (SMDR) records are generated for every call made. If other calls, such as Automatic Call Distribution (ACD) calls, are being activated at the same time, the register pegs accumulate for these events also.

Effect of commands on OMs

The command DEVSTART in the LNKUTIL Command Interpreter (CI) increment states the link and the type of transfer going across this link. It also implies that a DEVCON command has been used to successfully create a pool and a device.

DEVSTART enables the display of OM tuples in the SLLNK OM group for this pool and its report transfer type. The information field of this tuple includes the pool name and the report transfer type, followed by the three registers (SLLNKOVF, SLLNKOK, and SLLNKQU), with each register initially set to zero.

Data link status

If the data link status is altered to any other state other than "transferring", the tuple regarding this pool and its transfer type no longer displays (refer to the LNKSTAT command for displaying link status), and the incrementing of its registers ceases. The tuple information can be retrieved when the link starts transferring again.

A new OM tuple receives a new index when being added.

If the DEVDISC device KILL command removes the data link from existence along with the pool, the new OM tuple takes over its index when being added.

The user must be familiar with the transferring process to accumulate proper register counts. Refer to "Human-machine interface" (page 51) for information on the LNKUTIL and SMDRLNK CI increment commands.

Log reports

Log reports SLNK102 and SLNK103 pertain to Station Message Detail Recording (SMDR) reports data transfer.

SLNK102

The system generates log report SLNK102 for each data link device whenever the DEVSTART command in the LNKUTIL Command Interpreter (CI) increment starts data transfer. The system generates this log for each data link in the pool on which the POOLSTART command starts data transfer.

SLNK102 provides information only. No action is required.

The following example shows the report format for SLNK102:

```
SLNK102 APR22 06:45:22 1999 INFO SESSION
SMDR Reports transfer started on device
MRLINK.
```

SLNK103

The system generates log report SLNK103 for each data link device when data transfer stops on a data link using the DEVSTOP command in the LNKUTIL CI increment. The system also generates this log for each data link in the pool on which data transfer is stopped using the POOLSTOP command.

No action is taken if a user at the Maintenance and Administration Position (MAP) workstation stops the transfer. If no manual action is taken, the transfer terminates through software. Accompanying Software Error Report (SWERR), Trap, and logs provide further information.

The following example shows the report format for SLNK103:

```
SLNK103 APR22 06:45:22 1999 INFO SESSION
SMDR Reports transfer stopped on device
MRLINK.
```

List of terms

ACD	Automatic Call Distribution
ACDMR	Automatic Call Distribution Management Reports
AMA	Automatic Message Accounting
ASCII	American Standard Code for Information Interchange
BC	Bearer Capability
CCITT	International Telephone and Telegraph Consultative Committee
CI	Command Interpreter
DAO	Digits As Outpulsed
DIRP	Device Independent Recording Package
DSP	Down Stream Processor
HMI	Human-Machine Interface
IBN	Integrated Business Network

IOC	Input/Output Controller
IOM	Input/Output Module
IP	Internet Protocol
ITC	Inbound Toll Call
MAP workstation	Maintenance and Administration Position workstation
MSL	Meridian SL-100 system
OM	Operational Measurement
OPDU	Operation Protocol Data Unit
OSI	Open Systems Interconnect
RO	Remote Operation
SMDR	Station Message Detail Recording
SMDRITC	Station Message Detail Recording Inbound Toll Call
SWERR	Software Error Report
TCP	Transmission Control Protocol

Communication Server 2100

ASCII SMDR Data Access Description and Implementation

Publication: 555-4001-119
Document status: Standard
Document version: 07.01
Document date: 20 October 2006

To provide feedback or report a problem in this document, go to www.nortel.com/documentfeedback.

The information in this document is sourced in Canada, the United States of America, and the United Kingdom.