IBM Sterling Connect:Direct for OpenVMS

User's Guide

Version 3.6

This edition applies to the 3.6 Version of IBM® Sterling Connect:Direct® for OpenVMS and to all subsequent releases and modifications until otherwise indicated in new editions.

Before using this information and the product it supports, read the information in *Notices, on page121*.

Contents

Chapter 1

About Sterling Connect:Direct for OpenVMS

The IBM® Sterling Connect:Direct® product links technologies and moves all types of information between networked systems/computers. It manages high-performance transfers by providing such features as: automation, reliability, efficient use of resources, application integration, and ease of use. Sterling Connect:Direct software offers choices in communications protocols, hardware platforms, and operating systems. It provides the flexibility to move information among mainframes, midrange systems, desktop systems, and LAN-based workstations.

The Sterling Connect:Direct for OpenVMS product supports connectivity between OpenVMS systems and the following systems:

- IBM systems (MVS, VM, and VSE) using Systems Network Architecture (SNA) and Transmission Control Protocol/Internet Protocol (TCP/IP)
- VAX, MicroVAX, or Alpha AXP systems using DECnet and TCP/IP
- HP NonStop, UNIX, NT, and OS/400 systems using TCP/IP

Refer to the *IBM Sterling Connect:Direct for OpenVMS Release Notes* for enhancement and release-specific information.

Chapter 2

Understanding the User Interface

This chapter summarizes how to use Sterling Connect:Direct for OpenVMS through the user interface. The command syntax is also discussed.

Using Sterling Connect:Direct for OpenVMS

To use Sterling Connect:Direct for OpenVMS interactively, you must first invoke the user interface. You can invoke the user interface in one of three ways. Check with your Sterling Connect:Direct administrator to determine the method to use, based on the way the product is installed.

To run Sterling Connect:Direct for OpenVMS software:

- As a native DCL command, enter **NDMUI** at the command line. This option is available if your administrator installed the Sterling Connect:Direct product and added the user interface in the DCL command tables.
- As a foreign DCL command, you must define user interface as a foreign command by entering the following for each login:

```
$ NDMUI:==$NDM$$DIRECTORY:NDMUI
```

  Include the previous line in your LOGIN.COM file to avoid entering this string after each login.

- With the DCL **RUN** command, enter the following at the command line:

```
$ RUN NDM$$DIRECTORY:NDMUI
```

  This method does not support a single-line command, such as:

```
$ RUN NDM$$DIRECTORY:NDMUI SUBMIT process-name <CR>
```

Understanding the User Interface

The user interface is the interface through which you communicate with Sterling Connect:Direct. The modes of operation determine the way you communicate with the server through the user interface. You can execute all commands in either of the modes of operation as shown in the following examples:

- Noninteractive Mode

  Enter Sterling Connect:Direct commands, such as SUBMIT, at the OpenVMS prompt for noninteractive operation mode. **NDMUI** must precede all commands. After the command executes, control returns to the OpenVMS prompt.

```
$ NDMUI SUBMIT TODALLAS
```

- Interactive Mode

  You can invoke interactive mode by entering **NDMUI** at the DCL command-line prompt. The prompt changes to *CONNECT:Direct>*. You can then enter a series of Sterling Connect:Direct commands. Press the **RETURN** key at the end of each command to execute the command. The **EXIT** command returns control to DCL.

```
CONNECT:Direct> COPY SYS$LOGIN:LOGIN.COM -
_> SC.MVS.DALLAS"SMITH QWERTY"::SMITH.LOGIN.COM
CONNECT:Direct> EXIT
$
```

The user interface maintains an internal command recall buffer. By using the up-arrow and down-arrow keys on the keyboard, you can recall up to 20 previously issued commands. The command can be reissued as is, or it can be modified and reissued.

Understanding the Command Syntax

Sterling Connect:Direct for OpenVMS command syntax includes abbreviations, comments, and continuation marks.

Using Abbreviations

Abbreviations allow you to use as few keystrokes as possible to enter commands, required parameters, and qualifiers.

Commands can be abbreviated to the shortest unique length, as with DCL. For example, you can abbreviate the SHOW PROCESS command to SHO PROC, because no other Sterling Connect:Direct command begins with the letters *SHO* and *PROC*. Similarly, you can abbreviate the /OUTPUT qualifier as /OUT, because no other qualifier begins with *OUT*.

The following example shows the use of the SHO PROC and /OUT abbreviations:

```
$ NDMUI SHO PROC /PNAM=TODALLAS /OUT=DALLAS.LIS
```

Using Comments

Comments within Sterling Connect:Direct command procedures are denoted by asterisk (*). Command procedures are discussed in the @FILENAME Command section of Chapter 4, *Using Sterling Connect:Direct for OpenVMS Commands*. The following is a sample of commented text:

```
*    This Process copies the text files developed at
*    our San Francisco location to our data center in
*    Dallas.
*
*    This Process executes daily at 18:00 PST.
*
*    In case of emergency, contact the following persons in
*    the order in which they are listed.
```

Continuing a Command on Additional Lines

The hyphen (-) is used as a continuation mark in Sterling Connect:Direct commands. The continuation mark can be used at any time to split a command that is too long to fit on one line or as a break to make a lengthy command more readable. The continuation mark must be the last character of the line, with the continuation text entered after the prompt on the following line. There is no limit to the number of lines that you can use to enter a command, as long as you do not exceed the DCL command limit of 1,022 characters.

The following example shows how to continue a command in noninteractive mode.

```
$ NDMUI SUBMIT TODALLAS /hold=yes /startt=("",13:00:00) -
_>  /prty=7
```

The following example shows how to continue a command in interactive mode.

```
CONNECT:Direct> SUBMIT TODALLAS /hold=yes  -
_> /startt=("",13:00:00) /prty=7
```

Chapter 3

Using Additional Features

This chapter contains information about the following Sterling Connect:Direct for OpenVMS features:

- Logicals
- Type file records
- Security
- Checkpoint-restart
- Online messages
- Case sensitivity
- Remote procedure execution
- Script compilation
- Event logging facility

Using Sterling Connect:Direct for OpenVMS Logicals

The Sterling Connect:Direct for OpenVMS software requires that certain OpenVMS logical names be defined. During product installation, your system administrator should have set up all required logical names.

Define your default Process directory as NDM$$PROCESS. When a SUBMIT command is issued, if the Process is not found in the current directory, the directory defined as NDM$$PROCESS is searched.

Using Sterling Connect:Direct for OpenVMS Type File Records

Type file records are stored as entries in the NDM_TYPE.TLB text library. Entries include FIXED, FORTRAN, IMAGE, RELATIVE, SAVESET, STREAM, STREAM_CR, and STREAM_LF. Add your own type records into the library by creating a customized record and inserting it into the text library. Refer to the DEC *File Definition Language* (FDL) manual for further details. Online help within OpenVMS is also useful. To view online help, issue the following command:

```
$ HELP topic
```

Implementing Sterling Connect:Direct for OpenVMS Security

Login processing is performed automatically when you invoke the user interface or specify the /SERVER qualifier.

Validation of your authorization to sign on to a particular Sterling Connect:Direct for OpenVMS (including the local node) is performed by the server, using the OpenVMS user name used for the login. If the server node is different from the node running the user interface, the user name is treated as a proxy id on the OpenVMS node running the server. The user name is processed in a way similar to a proxy login initiated by DECnet.

If the user interface is running on the same node as the server, the User Authorization File (UAF) is checked to verify that the user name exists for that node; however, no password verification is performed.

Using Security Checking for Sterling Connect:Direct Processes

Security checking during Process execution is based on a user name in the Process submitter field and information in either the PNODEID or SNODEID fields, if they exist.

If Sterling Connect:Direct for OpenVMS is executing a Process, it uses the submitter id and PNODEID information, if any, for security checking. If the remote node is executing a Process, Sterling Connect:Direct for OpenVMS uses the submitter id and the SNODEID information, if any, for security checking.

The Sterling Connect:Direct for OpenVMS software validates the security id for the Process in one of two ways: by verifying a user name and password in the UAF or verifying that a node and user name combination for a particular remote node exists in the OpenVMS proxy database (NETUAF) in a way similar to a DECnet proxy login.

- If an OpenVMS user name and password are specified in the PNODEID or SNODEID parameters of a Process, Sterling Connect:Direct for OpenVMS verifies the user name and password in the UAF. The password is then converted into an internal OpenVMS format and is compared with the password field in the UAF record for the specified OpenVMS user name. If the password values do not match or the UAF record does not exist, the security check fails.
- Verifying that a node and user name combination exists in a proxy database has the advantage that passwords are not required in Processes. The Process submitter's user name is treated as an OpenVMS proxy id. You can specify an OpenVMS user name existing in the UAF in the PNODEID or SNODEID parameter. The OpenVMS user name is used as the security id, if the proxy id processing is successful.

Using Symbols to Check NDM Command Execution

Five symbols can be used within a command procedure to check Process status that is returned by the user interface. These symbols indicate the status of the last command performed by the user interface. If you want to verify that the user interface successfully executed a command, you can interpret the following symbols within your DCL command procedure to retrieve status information.

Symbols	Status
NDM$$FDBK	Feedback string value displayed on the status line
NDM$$MSGID	Message ID string displayed on the status line
NDM$$PNAME	Process name string displayed on the status line

Symbols	Status
NDM$$PNUM	Process number string displayed on the status line
NDM$$RC	Reply string value displayed on the status line

Note: The standard OpenVMS symbol $SEVERITY is also returned.

These symbols correspond to the status message displayed upon completion of a command issued in either interactive or noninteractive mode. An example of a status message for a Process named SEND_VB that submitted successfully follows:

```
VSRV101I:  Feedback:  0  Reply:   0   Function: dtf_submit
Process submitted successfully.   Process number  :  6
```

If this Process is submitted within your command procedure, the previous status message is assembled into the following status symbols:

Symbols	Corresponding Value
NDM$$FDBK	0
NDM$$MSGID	VSRV101I
NDM$$PNAME	SEND_VB
NDM$$PNUM	6
NDM$$RC	0

Note: You can display Sterling Connect:Direct symbols by issuing the following OpenVMS command:

SHOW SYMBOL NDM$$*

If a Process is not submitted successfully, a Process name and Process number is not assigned. Therefore, the values of the symbols representing PNAME and PNUMBER are NDM_NOPNAME and NDM_NOPNUM, respectively. If the operation does not have a status, the value of NDM$$MSGID is NDM_NOMSGID.

Using the Checkpoint-Restart Feature

The Sterling Connect:Direct for OpenVMS product provides a checkpoint-restart feature for sequential disk files. This feature eliminates the need to retransmit an entire file in the event of a transmission failure. A value on the COPY statement or in the initialization parameters specifies the checkpoint interval. If a COPY procedure is interrupted, the Sterling Connect:Direct software restarts that COPY operation at the last checkpoint.

Checkpoint-restart functions whether Sterling Connect:Direct for OpenVMS is the sender or the receiver. Checkpoint-restart is a function of the COPY statement, therefore, refer to the *IBM Sterling Connect:Direct Process Language Web site* for further information.

Viewing Online Messages

The Sterling Connect:Direct for OpenVMS product has an online message file, consisting of short text, followed by long text. Only the short text is displayed by the user interface when an error occurs.

Displaying Message Identifiers

You can use a DCL procedure to display Sterling Connect:Direct message identifiers. Use the following command to invoke the procedure:

```
@ ndm$msgid_display  msgid  [msgfile]
```

The following are definitions for the DCL procedure parameters:

msgid
is the message identifier.

msgfile
is the C:D OpenVMS message file.

If the message file parameter (p2) is not specified, the default value is **ndm$$directory:msgfile.dat**.

There is no default for the message identifier parameter (p1). If the message identifier does not exist, no diagnostic information is displayed.

Specifying Case Sensitivity

Some remote nodes that Sterling Connect:Direct for OpenVMS communicates with are case-sensitive operating systems. To preserve lowercase or mixed-case characters, you must enclose all filenames in quotation marks. To accommodate case-sensitivity requirements, the following qualifiers and commands are included as applicable.

To accommodate case sensitivity requirements, use the following qualifiers and commands:

- Qualifiers

  The /CASE qualifier ensures that case is preserved for all characters enclosed in quotation marks. If you specify the /NOCASE qualifier, Sterling Connect:Direct ignores the case of characters enclosed in quotation marks and converts them to uppercase. You can specify the /CASE and /NOCASE qualifiers with the following commands:

  - SUBMIT
  - CHANGE PROCESS
  - DELETE PROCESS
  - SELECT PROCESS
  - SUSPEND PROCESS
  - SELECT STATISTICS

- Commands

SET CASE defines a logical NDM$$PRESERVE_CASE. SET NOCASE clears the logical. There are no parameters or qualifiers for these commands.

Note: You can define the NDM$$PRESERVE_CASE logical in the process table outside of Sterling Connect:Direct.

If you define NDM$$PRESERVE_CASE, the /CASE qualifier becomes the default and need not be specified explicitly when issuing commands. To override the default, you must specify the /NOCASE qualifier when issuing commands. If you do not define NDM$$PRESERVE_CASE, the /NOCASE qualifier becomes the default.

If you define the NDM$$PRESERVE_CASE logical but specify /NOCASE, Sterling Connect:Direct converts all characters to uppercase. If you do not define the NDM$$PRESERVE_CASE logical but specify /CASE, Sterling Connect:Direct preserves case for all characters that Digital Command Language (DCL) does not convert, for example, characters enclosed in quotation marks.

Compiling a Script

By using a compiled script, you can:

- reduce the time-to-execution overhead
- increase security by concealing the remote access control stored in script files or in DCL procedures
- enhance your ability to use the native OpenVMS queuing and scheduling facilities
- use it repeatedly without modifying

Methods of Compiling a Script

To request script compilation, use the /CSO qualifier on the Sterling Connect:Direct for OpenVMS NDMUI Submit verb as shown in the following example:

```
$ ndmui  submit  script.ndm  /cso
```

The resulting compiled script is written to a file named **ndm.csx**. You can rename the file if desired.

Note: The /CSO qualifier does not execute the script; it just requests that a script object file be produced for later use.

This form of script execution is faster and less resource-intensive than using the user interface **(ndmui)** program. This is because no server process connection being is required and repetitive parsing is eliminated.

You can execute a compiled script directly within a DCL procedure or by the API routines. For information on executing a compiled script using the API, refer to the *CSX Application Programming Interface* section beginning on page 5-76.

Execute a compiled script within a DCL procedure by using the **ndm_csx_script_submit.exe** image in either of the following methods:

```
$ run  ndm$$directory:ndm_csx_script_submit.exe
```

```
$ run  ndm_csx_scipt_submit.exe
```

Script Compilation Parameters

Define the image parameters by symbol or by logical name. If the symbol is undefined, the corresponding logical names are translated. Currently, the only required parameter is the parameter that specifies the filename of the script to be submitted. All other parameters are optional. These parameters are as follows:

ndm$$csx_script_submit_filename
the filename of the script to be submitted. This parameter is required.

ndm$$csx_script_submit_environment
the optional environment name. The default value is NDM.

ndm$$csx_script_submit_waitcomplete
this optional definition specifies that the process is to await completion of the script. The value of this parameter is irrelevant. The default action is to proceed.

ndm$$csx_script_submit_waittimeout
this optional definition specifies the maximum time (in minutes) to await script completion. The default value is indefinite.

Note: This parameter is ignored if the **ndm$$csx_script_submit_waitcomplete** parameter is not specified.

Note: A compiled script cannot be the object of a Sterling Connect:Direct for OpenVMS Submit statement regardless of whether it is issued by the **ndmui** (user interface) program or whether it appears as a language statement in a script file.

If the compiled script is successfully submitted, the following primary symbols will be defined:

ndm$$csx_sx_rqid
the request identifier assigned to the script by Sterling Connect:Direct for OpenVMS

ndm$$csx_sx_name
the request name assigned to the script by the user.

ndm$$csx_sx_number
the request number assigned to the script by Sterling Connect:Direct for OpenVMS

If the **ndm$$csx_script_submit_waitcomplete** parameter is specified, you can define a symbol (**ndm$$csx_sx_term**) that reflects script termination status. Use the following Run Job clause to define the termination status:

```
run job  (dsn="#csx$status  term_status"     pnode)
```

In the previous example, *term_status* is a decimal ASCII string.

Double-quotes ("") must delimit the string and the pnode must be specified as part of the Run Job clause. An example follows:

```
csx_status  run job  (dsn="#csx$status 98962" pnode)
```

The previous script language statement returns a status of *file-not-found* to the process awaiting completion.

Usually the termination status returned to the process awaiting completion should be an OpenVMS completion code; however, this is not enforced. If you did not define this symbol, or if an unrecoverable error occurred

(such as a remote authorization failure), Sterling Connect:Direct for OpenVMS will define this symbol. If Sterling Connect:Direct for OpenVMS defines this symbol, it will always be a valid OpenVMS system completion code.

Completion status is not and cannot be returned in the same manner and context as in standard OpenVMS constructions and command procedures. Successful submission does not imply successful execution.

As a result, the primary symbols could be defined to reflect successful script submission, but the script execution may actually fail.

Although not required to produce a script object file, OPER privilege is required to execute a compiled script.

Using Placeholders

To allow a compiled script to serve as a template, you can use placeholder specifications.

A placeholder is specified by a three-character reserved word sequence (trigraph) symbol of the form **?xn** which is replaced by its value at execution time.

The trigraphs are as follows:

```
?p1  ?p2   ?p3  ?p4  ?p5  ?p6  ?p7  ?p8
?q1  ?q2   ?q3  ?q4  ?q5  ?q6  ?q7  ?q8
?r1  ?r2   ?r3  ?r4  ?r5  ?r6  ?r7  ?r8
?s1  ?s2   ?s3  ?s4  ?s5  ?s6  ?s7  ?s8
?t1  ?t2   ?t3  ?t4  ?t5  ?t6  ?t7  ?t8
```

An **&** can be used instead of an **?** and the symbol case is ignored. That is, **?p1** is equivalent to **&P1**.

This type of a pattern is replaced by its value at execution time. The value of an undefined symbol reference is a null string.

The value of a placeholder is obtained as follows (progressing in order of failure):

1. Translate the symbol name as a symbol.
2. Translate the symbol name as a logical name.
3. Translate the derived name as a logical name.
4. The placeholder value is null.

If Step (2) fails, a derived name of the form **ndm$$csx_$symbol$_[placeholder]** is translated as a logical name.

For example, if the name *q3* did not exist as either a symbol name or as a logical name, the logical name **ndm$$csx_$symbol$_q3** would be translated. If this logical name does not exist, the value of the *?q3* placeholder would be the zero-length (null) string.

The following sample script (sample.txn) and procedure fragment indicates one of several methods for generating unique filenames when transmitting to a remote node.

```
sessname  process  snode=?s1
txstep    copy     from (dsn=123.txt         pnode)-
                   to   (dsn=hlq.?q1.?q2  snode)
```

The placeholder symbols are defined by a DCL procedure as follows:

```
$!
$!      Lookup Remote Node Name
$!
$  s1  = "sci.remote.node"
$!
$!  Generate Filename Identifiers
$!
$  tm = f$cvtime ( "" )
$  q1 = f$extract (  0, 4, tm )   +               -
        f$extract (  5, 2, tm )   +               -
        f$extract (  8, 2, tm )
$  q2 = f$extract ( 11, 2, tm )   +                -
        f$extract ( 14, 2, tm )   +                -
        f$extract ( 17, 2, tm )   +                -
        f$extract ( 20, 2, tm )
$!
$!  Issue Request to C:D OpenVMS
$!
$  ndm$$csx_submit_filename = "sample.txn"
$  run ndm$$directory:ndm_csx_script_submit
$  if  .not.  $status
$  then
$     say "C:D Submit Failure"
$  endif
$  exit $status
```

A placeholder can be specified anywhere in the source script that a syntactically valid token may appear, as in the following example:

```
?p1      process  snode=?p2      snodeid=?r1
?s1      copy     from (dsn =  ?t1     pnode ) -
                  to   (dsn =  ?t2     snode )
?s2      run job       (dsn = "?t3"    snode )
```

In this example, the script field names are determined as follows:

```
 - script name                      P1
 - remote node name                 P2
 - remote user name                 R1
 - name of COPY step                S1
 - name of source file              T1
 - name of destination file         T2
 - name of Run Job Step             S2
 - name of Run Job procedure file   T3
```

Access Control Considerations

A placeholder specifier has no meaning, and its text is substituted where it appears. The exception to this is the **?_actl** (access control) reserved word placeholder.

In the following example, the remote access control is always given by the value of the **?q1** placeholder and no password can be specified:

```
txn00001   process  snode=remote.node   snodeid=(?q1)
```

In the following example, the remote access control of the compiled script is constant.

```
txn00002   process  snode=?s1          snodeid=(usr,pwd)
```

To specify remote access control in a more flexible manner, the **?_actl** placeholder should be specified as in the following example:

```
txn00003  process  snode=remote.node    snodeid=(?_actl)
```

This placeholder (or alternatively, **&_actl**) is used exclusively for this purpose and cannot appear more than once in a compiled script. Unlike other reserved word placeholder specifiers, the placeholder value is subject to further processing.

Specifically, the character sequences **%#**, and **%@** are considered to be introducers when they appear at the at the beginning of an access control placeholder value.

The access control placeholder allows you to specify remote access control directly, indirectly, immediately, or not at all in the following forms:

- A) Direct username
- B) Indirect %@access_control_identifier
- C) Immediate %#username:password
- D) NULL NULL

In (A) the effective specification is as follows:

```
txn00001  process  snode=remote.node  snodeid=(usr)
```

If the introducers appear as part of the actual username, you can use a script with another placeholder specifier or multiple scripts each with constant remote access control.

In (B) the text following the **%@** is considered to be an access control identifier used to retrieve the remote username and encrypted password from a previously generated password file. The effective specification is as follows:

```
txn00001 process snode=remote.node  snodeid=(usr, pwd)
```

If the identifier is undefined or the password file does not exist, this condition is processed as if no access control had been specified.

In (C) the text following the **%#** is considered to be a plain-text username and password separated by a **:** (colon). This form will be used by programs that define the remote access control placeholder, but which do not make this information ordinarily visible. The password is then encrypted as if it had been directly specified. The effective specification is as follows:

```
txn00001 process snode=remote.node  snodeid=(usr,pwd)
```

In (D) the placeholder is specified but is undefined. This case is treated as if no access control were specified at all. The effective specification is as follows:

```
txn00001 process snode=remote.node
```

The following is a placeholder example.

```
$!
$! [Direct]   Specify The Remote Username
$!
$ _actl = "usr"         ! Specify Remote Username

$!
$! [Indirect] Retrieve Username/Password Of
$!            "RemoteNode::RemoteUser" Entry
$!
$ _actl = "%@RemoteNode::RemoteUser"
$!
$! [Immediate] Specify Username/Password Explicitly
$!
$ _actl = "%#usr:pwd"
$!
$! [NULL]     Remove All Remote Access Control
$!
$  delete /symbol  _actl
```

Note: These considerations do not apply to the PNODEID clause nor do they apply to SNODEID clauses of the form **snodeid=(usr,pwd,new)**.

Using reserved word placeholders is optional. You can compile a script to conceal the access control of a remote system and then archive the original script file. In this case, security is enhanced because plain-text passwords are not present in script files. By using the password file, security can be further enhanced since remote plain-text passwords do not need to be present in script files or in DCL procedures.

Using the Event Logging Facility

The event logging facility allows you to obtain near real-time status messages from Sterling Connect:Direct for OpenVMS. You can process these messages, site-written programs, and procedures, thus eliminating the need to use resource-intensive statistics using keyed files. As a result, event logging may increase operational efficiency by:

- Eliminating the overhead associated with keyed files and foreign keys
  - The optional event log is a sequential file.
- Minimizing internal data conversions
  - Conversions are performed only when needed.
- Reducing the amount of information generated along with more concise content.
- Facilitating the promulgating of information by site.

The event logging component of Sterling Connect:Direct for OpenVMS is separate from the rest of the product. As a result, you can install it on a node where Sterling Connect:Direct for OpenVMS is not installed. The event process can then function as a network sink agent for a cluster or other network nodes.

Enabling and Disabling Event Logging

If event logging resides on a system where Sterling Connect:Direct for OpenVMS is installed, its startup procedure must be executed before the startup procedure of Sterling Connect:Direct for OpenVMS. If Event Logging will not be used, no changes to the system startup procedure are required.

Use the following event process startup and shutdown procedures to enable event logging:

```
@ ndm$event_process_startup
@ ndm$event_process_shutdown
```

The event process startup parameter file (ndm$event_process_setparams.com) describes the startup parameters. The ndm$$event_process_log_file_startup_disable parameter disables event message recording.

You may want to disable event message recording for the following situations:

- to forward generated messages to a remote sink node
- to place such functionality into site-written event procedures
- to process events with site-written programs that use the event API

Event messages are logged in files that can only be displayed directly through site-written programs or indirectly through site-written event procedures.

Statistics files recording is still supported, however, you should disable this function when using the event logging facility. This will further reduce any unnecessary system overhead. To disable the recording of statistics files, specify the following parameter in the initial parameter file (initparms.dat):

```
ndm$$stat_shutdown      y
```

Event Message Format

Events are identified as an event class and an event type. The class denotes the major component definition and the type specifies the event within that component class.

The event logs are variable length sequential files with a maximum record length of 2048 bytes.

Note: The format of the event log files will not change in future releases of Sterling Connect:Direct for OpenVMS.

Each record/message in a Sterling Connect:Direct for OpenVMS event log/stream has the following format:

```
0            8           16          18       20       24           28           32
-------------------------------------------------------------------------------------
Event Time  | Reserved   | Event     | Event  | Event  | Event      | Reserved   | Event
            | (SBZ)      | Class     | Type   | Node   | Flags      | (SBZ)      | Data
```

The field (decimal) displacement is displayed above each field. The following table lists descriptions for each field.

Field	Description
Event Time	A signed quadword that specifies the source node local absolute system time of the event.
Reserved (1)	An unsigned quadword reserved for future use.
Event Class	An unsigned word which specifies the class of the event.
Event Type	An unsigned word which specifies the type of the event.
Event Node	An unsigned longword which specifies the site-specified source node context value of the event. The value **0** is reserved.

Field	Description
Event Flags	An unsigned flags longword used by the Sterling Connect:Direct for OpenVMS event facility.
Reserved (2)	An unsigned quadword reserved for future use.
Event Data	The event message data in binary [itemcode,value] format. The length of the event message data is the message/record record length less 32.

The following are descriptions of the event message/record format in selected languages:

```
*  Macro                          .psect  event_message,abs
                  event_time:      .blkq   1
                  reserved_1:      .blkq   1
                  event_class:     .blkw   1
                  event_type:      .blkw   1
                  event_node:      .blkl   1
                  event_flags:     .blkl   1
                  reserved_2:      .blkl   1
                  event_data:      .blkb  2048-event_data-event_time

*  Fortran        STRUCTURE  /EVENT_MESSAGE/
                      CHARACTER*8      EVENT_TIME
                      CHARACTER*8      RESERVED_1
                      INTEGER*2        EVENT_CLASS
                      INTEGER*2        EVENT_TYPE
                      INTEGER*4        EVENT_NODE
                      INTEGER*4        EVENT_FLAGS
                      INTEGER*4        RESERVED_2
                      CHARACTER*2016   EVENT_DATA
                  END STRUCTURE

*  C              structure  event_message
                             {
                               unsigned   int   event_time  [  2  ] ;
                               unsigned   char  reserved_1  [  8  ] ;
                               unsigned   short event_class         ;
                               unsigned   short event_type          ;
                               unsigned   int   event_node          ;
                               unsigned   int   event_flags         ;
                               unsigned   int   reserved_2          ;
                               unsigned   char  event_data  [2016 ] ;
                             }                                      ;
```

Local systems can receive event messages through site-written event procedures and event stream listener programs.

Event Procedures

A site-written procedure that executes whenever a particular event occurs is termed an event procedure. The content of an event message is specified to an event procedure in such a way that it is easily processed by a DCL. An event procedure driver process is supplied to assist in writing site-written event procedures. The event procedure driver process is executed as follows:

```
@  ndm$event_procedure_startup
```

The input to this procedure is a parameter template file as follows:

```
$!
$! C:D OpenVMS Event Procedure Parameter Template File
$!
$!define ndm$$event_procedure_filename            procedure_filename
$!
$!                 This parameter defines the procedure to be
$!                 executed when an event occurs and must be defined.
$!
$!                 Default: None
$!
$!define ndm$$event_procedure_directory      procedure_directory_name
$!
$!                 This parameter defines the directory from which
$!                 the constructed procedures will be executed.
$!
$!                 Default: None
$!
$!define ndm$$event_procedure_queue               sys$batch
$!
$!                 This parameter defines the queue to which the
$!                 procedure is to be submitted for execution.
$!
$!                 Default: SYS$BATCH
$!
$!define ndm$$event_procedure_queue_001           queue_001
$!                  ...
$!define ndm$$event_procedure_queue_255           queue_255
$!
$!                 These parameters define additional queues to which
$!                 procedures may be submitted for execution in turn.
$!
$!                 Default: None
$!
$!define ndm$$event_procedure_queue_username      startup_username
$!
$!                 This parameter defines the username under which
$!                 the procedure is to be submitted for execution.
$!
$!                 Default: The event procedure process username.
$!
$!define ndm$$event_procedure_logfile             Y
$!
$!                 This parameter specifies that a procedure execution
$!                 log file is to be generated and is the equivalent
$!                 of /LOG. This parameter would normally be defined
$!                 externally to the Event Procedure process startup
$!                 and then deassigned when procedure log files are no
$!                 longer desired.
$!
$!                 Default: /NOLOG
$!
$!define ndm$$event_procedure_priority            6
$!
$!                 This parameter defines the priority at which
$!                 the event procedure process executes.
$!
$!                 Default: 6
$!
```

(continued)

```
$!define ndm$$event_procedure_keep               Y
$!
$!                 This parameter specifies that the event context
$!                 procedure file is to be retained after the event
$!                 procedure terminates. This logical name is dynamic
$!                 and may be defined and deassigned as desired.
$!$!                 Default: The event context procedure is deleted.
$!
$!define ndm$$event_procedure_trigger_CCCTTT      Y
$!                 ...
$!define ndm$$event_procedure_trigger_001002      Y
$!
$!                 These parameters define the events which trigger
$!                 the execution of the event procedure. Only those
$!                 events defined as triggers will cause the event
$!                 procedure to be executed.
$!
$!                 Default: All events are trigger events if no
$!                          definitions are specified.
$!
$!define ndm$$event_procedure_trigger$filter      Y
$!
$!                 This parameter specifies that event trigger names
$!                 actually specify event filter names. That is, only
$!                 those events NOT defined in the event list will
$!                 cause the event procedure to be executed.
$!
$!                 Default: The logical names specify an event trigger
$!                          list as opposed to an event filter list.
$!
$!define ndm$$event_procedure_noreference_list    Y
$!
$!                 This parameter specifies that no event data item
$!                 reference list is to be generated by the driver
$!                 procedure.
$!
$!                 Default: An event data item list of the form
$!                          'NDMEVL$EVENT_DATA_nn' is generated.
$!
$!define ndm$$event_procedure_item_description    Y
$!
$!                 This parameter specifies that a symbol of the form
$!                 'ITEM_NAME'_ specifying the description of
$!                 'ITEM_NAME' is to be generated for each event data
$!                 item.
$!
$!                 Default: No data item description is generated.
```

The event procedure driver process performs the following actions when an event message is received:

1. The list of event procedure triggers (event filter) is examined; if the event message does not pass the filter, it is discarded.

   The event procedure triggers (event filter) are listed in a specific form to expedite processing. For example, event 13.8 would be written as 013008. This format is used to specify the **P1** parameter when invoking the event procedure.

2. If the event passes the filter, a driver procedure is created in the directory specified by the ndm$$event_procedure_directory parameter. This in turn invokes the site-written event procedure specified by the ndm$$event_procedure_filename parameter.

The event procedure is invoked as follows:

```
@ ndm$$event_procedure_filename  event  time  count  node  sequence  [replay]
```

The following are the definitions for the previous parameters:

event—the event class/type in CCCTTT format

time—the event time in DD-MMM-YYYY HH:MM:SS.CC format

count—the number of event data items associated with the message

node—the site-specified node context unsigned longword value

sequence—the sequence number of this message assigned by the driver process

[replay]—event replay indicator: if present, the message is being replayed

The **P3** argument to the event procedure specifies the number of data items present in the message. The event data is decoded for use by the event procedure as a series of symbols given by the item name and value.

For example, the event procedure driver process represents the ^X70000020 item code in the driver context procedure as follows:

```
   $ NDMEVL$I$S_REMOTENODENAME="remote.node.name"
```

As a result, the event procedure can determine whether a particular data item (a step name) is present in the message as in the following examples:

```
$ if  f$type ( NDMEVL$I$S_SESSIONSCRIPTSTEP )  .eqs.  ""
$ then
$   write sys$output "Step Transaction Name/Number Not Present"
$ else
$   write sys$output  "Transaction Number: ", NDMEVL$I$S_SESSIONSCRIPTSTEP
$ endif
$ if  f$type ( NDMEVL$I$S_REMOTENODEMSGID )  .eqs.  ""
$ then
$   write sys$output "No Remote C:D Message Identifier Found."
$ else
$   write sys$output  "** Displaying Remote C:D Message Identifier **"
$   @ ndm$msgid_display  'NDMEVL$I$S_REMOTENODEMSGID
$ endif
```

In addition, a reference symbol is defined for each data item symbol so that the item list can easily be displayed, as follows:

```
   $ ndmevl$event_data_[k] = "name of kth data item"
                           ...
   $ ndmevl$event_data_[count]="name of last item"
```

For example, if the remote node name appeared as the fourth data item and the step name as the eleventh data item, the reference symbols would be defined as follows:

```
                           ...
   $ ndmevl$event_data_4 = "NDMEVL$I$S_REMOTENODENAME"
                           ...
   $ ndmevl$event_data_11 = "NDMEVL$I$S_SESSIONSCRIPTSTEP"
                           ...
```

The symbol ndmevl$event_description is also defined to specify the descriptive name/text associated with the event.

1. The created driver procedure is submitted to the next queue based on:
   - the values of the ndm$$event_procedure_queue
   - the ndm$$event_procedure_queue_nnn parameters under the username based on the value of the ndm$$event_procedure_queue_username parameter

   The ndm$$event_procedure_logfile, which can be defined at any time, is translated. If defined, a procedure log file is specified.
2. The created driver procedure executes.
3. The created driver procedure file is deleted unless the ndm$$event_procedure_keep parameter is defined.

For information on processing event messages using the API, refer to the *Event Application Programming Interface* section beginning on page 5-83.

Replaying Event Messages

You might encounter a situation where event messages are recorded, but the site-written event procedures (as well as any associated downstream procedures) did not execute properly. This could occur because of any number of technical reasons including cluster node failure, quota exhaustion, DCL coding errors, and so forth. As a precaution, you may want to process any outstanding events as part of the error recovery process.

Event messages can be replayed from an event log file as follows:

```
@  ndm$event_log_replay  logfile  after  before  remote
```

The following are definitions for the previous fields:

logfile
specifies the filename of the event log to be replayed. The default is None.

after
specifies the beginning absolute time. The default is Today.

before
specifies the ending absolute time. The default is Tomorrow.

remote
specifies if messages are broadcast to remote nodes. The default is No Broadcast.

Note: Event messages received in this manner are received with the replay indicator set. The **P6** (replay) event procedure argument would be set to REPLAY. Otherwise, the **P6** argument is not specified.

To determine if events would be replayed in a given time interval, without actually replaying them, define a logical name as follows:

```
$ define  ndm$$event_log_replay_readonly   y
```

In the previous example, the events between the times will be displayed, but not replayed.

Chapter 4

Using Sterling Connect:Direct for OpenVMS Commands

This chapter describes Sterling Connect:Direct for OpenVMS command functions, syntax, parameters, and qualifiers. Examples of the commands are provided at the end of each command section.

Some commands have parameters and qualifiers associated with them. Other commands have either parameters or qualifiers, but not both. Qualifiers are indicated with a forward slash character (/).

Reviewing Sterling Connect:Direct for OpenVMS Commands

Activities in Sterling Connect:Direct for OpenVMS are implemented with commands and Processes.

Modifying Processes

Use the CHANGE PROCESS command to modify Process priority and queue status or to change such job characteristics as destination node, start time, and start date of a *nonexecuting* Process.

Reviewing the Command Format

The following shows the command format and qualifiers for the CHANGE PROCESS command. Required qualifiers are in bold print. Default values for qualifiers are underlined. A description of each qualifier follows the CHANGE PROCESS command format. There are no parameters.

Command	Qualifiers
CHANGE PROCESS	/[NO]CASE
	†/AFTER
	/DEST=destination_nodename
	/HOLD=<u>No</u> | Yes

† Specify this time-based qualifier in the same format as other standard OpenVMS commands.

Command	Qualifiers
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/PRTY=number
	/RELEASE
	/SERVER=server_alias
	/STARTT=([date|day][,hh:mm:ssXM])
	/SUBMITTER=("submitter_node,submitter_id")|(list)

† Specify this time-based qualifier in the same format as other standard OpenVMS commands.

Parameters

There are no parameters associated with the CHANGE PROCESS command.

Required Qualifiers

You must specify one or more of the following qualifiers: **/DEST**, **/HOLD**, **/PRTY**, **/RELEASE**, **/SERVER**, or **/STARTT**.

If **/PNAME**, **/PNUMBER**, or **/SUBMITTER** are not specified, then the command will act on all processes that are accessible by the user.

Qualifiers

/[NO]CASE
allows for case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/DEST=destination_nodename
specifies a new destination node and changes the name of the node in the TCQ.

destination_nodename is the name of the destination node. Nodes that you can communicate with are listed in your network map.

/HOLD=No|Yes
delays the execution of a nonexecuting Process in the TCQ, releases a held Process for execution, or delays execution until *called*.

When you specify both HOLD=YES and a STARTT (start time and/or start date) value, the Process is placed in the Hold queue. If start time has not passed once you release the Process, the Process begins at the specified start time.

No (default) specifies that the Process is executed immediately.

Yes specifies that the Process remains in the Hold queue until you take one of the following actions:

- Release the Process with the CHANGE PROCESS /HOLD=NO command.

- Remove the Process from the TCQ with the DELETE PROCESS command.
- Release the Process for execution with the CHANGE PROCESS /RELEASE command.

/PNAME=name|(list)
searches for the Process by Process name. The length of Process names can range from 1–8 alphanumeric characters.

name specifies the name of a specific Process.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER=number|(list)
searches for the Process by Process number.

number specifies the Process number. Process numbers are assigned by Sterling Connect:Direct when Processes are submitted successfully. Process numbers can range from 1–99999.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/PRTY=number
allows you to change the priority of a Process in the TCQ. Priority is used for Process selection and does not affect the OpenVMS priority during transmission.

number is an integer ranging from 0–15.

/RELEASE
allows you to release a Process that has been suspended or held, that is, a Process in the Hold queue.

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

/STARTT=([date|day][,hh:mm:ssXM])
allows you to change the start time and either the day or date of a nonexecuting Process in the TCQ. Note that *date*, *day,* and *time* are positional; therefore, if the date or day is not specified, a null string and a comma must precede time. For example, the following translates to the Process running at 8:00 a.m. on the current date:

```
/STARTT("",8:00:00AM)
```

Day, date, and time are handled as a unit, so a change to one will affect the other two. Therefore, when you change the date, day, or time, you also must specify the information that you do not change.

date ensures that the Process executes on the specified date. You can specify dates in either Gregorian or Julian format.

Gregorian dates always include numeric values for the month (m), day (d), and year (y). Sterling Connect:Direct for OpenVMS only accepts the month-day-year format for Gregorian dates.

Enter Gregorian dates with either a two-digit or a four-digit year. You can use periods and backslashes (/) to separate the values.

Note: You must use a separator to specify a four-digit Gregorian year.

- mmddyy
- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

You can use periods and backslashes (/) to separate the values. Sterling Connect:Direct for OpenVMS processes the following Julian date formats:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If you specify only the date, time defaults to 00:00.

Note: If you are using a date format with slashes, start values should be enclosed within double quotation marks; for example:

/STARTT=("09/21/1998")

day releases the Process for execution on the specified day of the week.

If you specify a day of the week and you have used /RETAIN=YES, then the Process executes the same day every week.

You can specify **TODAY**, which releases the Process for execution today, or **TOMORROW**, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM. You do not have to specify minutes and seconds.

You can express time using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00 through 24:00. If you use the 12-hour clock, you must express time using AM or PM; therefore, the 24-hour clock is assumed if you do not use AM or PM.

If you specify hh:mm:ssXM and you use /RETAIN=YES, then the Process executes the same time every day.

You can specify **NOON**, which releases the Process for execution at noon, or **MIDNIGHT**, which releases the Process for execution at midnight.

/SUBMITTER=("submitter_node,submitter_id")|(list)
searches for the Process by the submitter node and submitter id of the Process submitter.

submitter_node is a 1–16 alphanumeric character name that specifies the symbolic node name of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the Process submitter.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

Examples

The following commands are examples of the CHANGE PROCESS command in noninteractive mode.

The following command changes the Process named TEST1 (in the Wait queue) from executing only once to executing every time the server is initialized:

```
$ NDMUI CHANGE PROCESS /PNAME=TEST1 /RETAIN=INITIAL
```

The following command changes the Process named TEST2 (in the Wait queue) to a new destination node (SNODE) of SC.VMS.VXNED1:

```
$ NDMUI CHANGE PROCESS /PNAME=TEST2 /DEST=SC.VMS.VXNED1
```

The following command changes Process number **3** named TEST3 on the Wait queue to start at 8:00 p.m. on Tuesday:

```
$ NDMUI CHANGE PROCESS /PNUMBER=3 /PNAME=TEST3 -
_> $/STARTT=(Tuesday,8:00PM)
```

Deleting Processes

The DELETE PROCESS command allows you to remove **nonexecuting** Processes from the TCQ.

Reviewing the Command Format

The following shows the command format and qualifiers for the DELETE PROCESS command. A description of each qualifier follows the DELETE PROCESS command format. There are no parameters.

Command	Qualifiers
DELETE PROCESS	/[NO]CASE
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/SUBMITTER=("submitter_node,submitter_id")|(list)
	/SERVER=server_alias

Parameters

There are no parameters associated with the DELETE PROCESS command.

If **/PNAME**, **/PNUMBER**, or **/SUBMITTER** are not specified, then the command will act on all processes that are accessible by the user.

Qualifiers

/[NO]CASE
allows for the case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/PNAME=name|(list)
searches for the Process by Process name.

name specifies the name of a specific Process. Process names can be from 1–8 alphanumeric characters long.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER=number|(list)
searches for the Process by Process number. Process numbers can range from 1–99999.

number specifies the specific number of a Process. Process numbers are assigned by Sterling Connect:Direct when Processes are submitted successfully.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/SUBMITTER=("submitter_node,submitter_id")|(list)
searches for Processes by submitter node and submitter id of the Process submitter.

submitter_node is a 1–16 alphanumeric character name that specifies the symbolic node name of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the Process submitter.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

Examples

The following examples illustrate the use of DELETE PROCESS commands in noninteractive mode.

The following command deletes a nonexecuting Process named TEST1 from the TCQ:

```
$ NDMUI DELETE PROCESS /PNAME=TEST1
```

The following command deletes the nonexecuting Process number 3 from the TCQ:

```
$ NDMUI DELETE PROCESS /PNUMBER=3
```

The following command deletes all nonexecuting Processes submitted by nodename SC.VMS.QA4 with a user id of QA11 from the TCQ:

```
$ NDMUI DELETE PROCESS /SUBMITTER=("SC.VMS.QA4,QA11")
```

Exiting Interactive Mode

Use the EXIT command when you are working in interactive mode and want to return to the DCL prompt. EXIT is equivalent to reading an end-of-file marker. Once in interactive mode, type **EXIT** or enter **Ctrl-Z**.

Reviewing the Command Format

The following shows the format of the EXIT command. The required information is in bold type. There are no parameters are qualifiers.

Command
EXIT

Parameters

There are no parameters associated with the EXIT command.

Qualifiers

There are no required qualifiers for the EXIT command

Example

Executing the EXIT command returns you to the DCL command-line prompt from interactive mode.

```
CONNECT:Direct> EXIT
```

Flushing an Executing Process

The FLUSH PROCESS command allows you to delete an *executing* Process from the TCQ.

Reviewing the Command Format

The following shows the format of the FLUSH PROCESS command.

The following shows the command format and qualifiers for the FLUSH PROCESS command. Required qualifiers are in bold print. A description of each qualifier follows the FLUSH PROCESS command format. There are no parameters.

Command	Qualifiers
FLUSH PROCESS	/FORCE
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/SUBMITTER=("submitter_node,submitter_id")|(list)
	/SERVER=server_alias

Parameters

There are no parameters associated with the FLUSH PROCESS command.

If **/PNAME**, **/PNUMBER**, or **/SUBMITTER** are not specified, then the command will act on all processes that are accessible by the user.

Qualifiers

/FORCE
flushes the Process by stopping the session manager. This qualifier is useful if a Process is in an unsatisfied wait state (hanging condition).

/PNAME=name|(list)
searches for the Process by Process name.

name specifies the name of a specific Process. Process names can range from 1–8 alphanumeric characters.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER= number|(list)
searches for the Process by Process number.

number specifies the specific number of a Process. The Process number is assigned by Sterling Connect:Direct when Processes are successfully submitted. Process numbers can range from 1–99999.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/SERVER= server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

/SUBMITTER=(“submitter_node,submitter_id”)|(list)
searches for the Process by the submitter node and submitter id of the Process submitter.

submitter_node is a 1–16 alphanumeric character name that specifies the symbolic node name of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the Process submitter.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

Examples

The following are examples of using the FLUSH PROCESS command in noninteractive mode:

The following command flushes an executing Process named TEST1 from the TCQ:

```
$ NDMUI FLUSH PROCESS /PNAME=TEST1
```

The following command flushes an executing Process number 3 from the TCQ:

```
$ NDMUI FLUSH PROCESS /PNUMBER=3
```

The following command flushes an executing Process submitted by nodename SC.VMS.QA4 user id QA11 from the TCQ:

```
$ NDMUI FLUSH PROCESS /SUBMITTER=("SC.VMS.QA4,QA11")
```

The following command flushes a hung Process named TEST1 from the TCQ:

```
$ NDMUI FLUSH PROCESS /PNAME=TEST1 /FORCE
```

Displaying Online HELP

The HELP command allows you to display online help information about Sterling Connect:Direct. For instance, you can display definitions and syntax requirements for any of the Sterling Connect:Direct for OpenVMS commands. Additionally, help information is available on a variety of subjects, including qualifiers, logicals, and file specifications.

The Sterling Connect:Direct HELP command is structured like the OpenVMS HELP command. You can use the HELP command either at the DCL noninteractive prompt or at the Sterling Connect:Direct prompt (interactive mode). Entering the HELP command without a topic displays a list of topics from which to choose. However, you can go directly to help information on a particular subject by entering the HELP command and the topic.

Reviewing the Command Format

The following shows the format for the HELP command. There are no qualifiers.

Command	Parameters
HELP	

Examples

This command displays a list of topics for which help is available.

```
$ NDMUI  HELP
```

Note: If the Sterling Connect:Direct for OpenVMS Help File was copied into the OpenVMS Help Library at installation, then Help can also be accessed through the OpenVMS Help Facility under the topic NDMUI. For example, the following command will display help about the SUBMIT command:

$ HELP NDMUI SUBMIT

The following figure shows an example of the Help Main Topic Screen. This screen displays when you request **HELP**.

```
NDMUI
     Invokes the User Interface

     NDMUI can operate under two modes: Interactive Mode and
     Noninteractive Mode.  Under Noninteractive Mode, NDMUI subcommands
     are entered via DCL prompt parameters and qualifiers.  If no
     parameters are supplied when invoking NDMUI, then NDMUI is invoked
     under Interactive Mode.  Under Interactive Mode, NDMUI subcommands
     are read from input (SYS$INPUT) until reaching end-of-file.  If
     SYS$INPUT is your terminal, you will see the prompt
     "CONNECT:Direct>" while you are running NDMUI.


Additional information available:

     SUBMIT   STOPNDM   SPAWN   EXIT   Statistics_Command
     Process Commands   Netmap Commands

NDMUI Subtopic?
```

The following figure shows an example of displaying a subtopic from the Help Topic Screen. This screen is obtained by entering the subtopic STOPNDM at the prompt on the Help Main Topic Screen.

```
NDMUI
     STOPNDM

     The STOPNDM command stops the server currently
     connected to the NDMUI.

     Format:
     NDMUI STOPNDM


Additional information available:

     Command qualifiers
     /FORCE    /IMMEDIATE    /QUIESCE    /STEP


NDMUI STOPNDM Subtopic?
```

By using the OpenVMS Help Facility, you can bypass the screens shown in the previous examples and display help information on a particular subtopic. The following figure shows an example of what you would type at the command line to get help information on the subtopic qualifier /FORCE for STOPNDM.

```
   $ HELP NDMUI STOPNDM /FORCE
```

The following figure shows the output from the previous command.

```
NDMUI

     STOPNDM

        /FORCE

           Stops CONNECT:Direct immediately by terminating
           the VAX/VMS process controlling CONNECT:Direct for
           OpenVMS.
```

Changing the Default Server

The SET SERVER command changes the default server accessed by the user interface.

Reviewing the Command Format

The following shows the command format and parameters for the SET SERVER command. Required parameters are in bold print. A description of each parameter follows the SET SERVER command format. There are no qualifiers.

Command	Parameters
SET SERVER	server_alias

Required Parameters

server_alias
specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

Note: To set the default server back to the local server, specify the server_alias parameter to be the local server environment name-NDM, for example, in a production environment.

Qualifiers

There are no qualifiers associated with the SET SERVER command.

Displaying the Process Last Submitted

The SHOW LAST command allows you to determine the Process number of the last submitted Process.

Reviewing the Command Format

The following is the format for the SHOW LAST command. There are no parameters or qualifiers.

Command
SHOW LAST

Parameters

There are no parameters associated with the SHOW LAST command.

Qualifiers

There are no qualifiers associated with the SHOW LAST command.

Examples

This example shows an output from the SHOW LAST command.

```
SHOW LAST
Last Assigned Process Number: 12
```

Displaying the Long Text of a Message

The SHOW MESSAGE command allows you to display the long text of a Sterling Connect:Direct message.

Reviewing the Command Format

The following shows the format and parameters of the SHOW MESSAGE command. The required parameter is in bold print. There are no qualifiers.

Command	Parameters
SHOW MESSAGE	message_id

Required Parameter

message_id
is the 8-character alphanumeric message id.

Qualifiers

There are no qualifiers associated with the SHOW MESSAGE command.

Listing Nodes in the Network Map

Use the SHOW NETMAP command to display a list of the nodes in your network map. You can determine which nodes you are authorized to use from the display. You can direct the output from the SHOW NETMAP command to a screen, printer, or file.

The network map file contains definitions for the local node, adjacent nodes, and passive connects. The local node is the node on which your OpenVMS process is running. The adjacent node is the node definition for a remote node that is used when the local node requests a connection. The passive connect record is the link from a remote node to the local node when the remote node requests the connection. Passive connects are used for the SNA communication connections only.

Refer to the *IBM Sterling Connect:Direct for OpenVMS Installation Guide* for the commands to alter the network map file.

Note: You can substitute SELECT NETMAP for SHOW NETMAP. **CTRL C** terminates output from all SELECT/SHOW commands.

Reviewing the Command Format

The following shows the command format and qualifiers for the SHOW NETMAP command. A description of each qualifier follows the SHOW NETMAP command format. There are no parameters.

Command	Qualifiers
SHOW NETMAP	/LOCAL_NODE
	/NODE=nodename|*|(list)
	/OUTPUT=filename
	/PASSIVE_CONNECT=name|*|(list)
	/PRINT
	/SERVER=server_alias

Parameters

There are no parameters associated with the SHOW NETMAP command.

Qualifiers

If you do not specify any of the following qualifiers, all are shown:

- /LOCAL_NODE
- /NODE
- /PASSIVE_CONNECT

/LOCAL_NODE
is the node on which your OpenVMS process is running; it is not necessary to refer to the nodename of the local node.

/NODE=nodename|*|(list)
is an adjacent node as defined in your network map.

nodename is a 1–16 character name of an adjacent node.

Asterisk (*) is a wildcard character used to indicate the names of all nodes.

You can specify a list of node names. The list is enclosed in parentheses, and each node name is separated by a comma; for example:

```
/NODE=(node1,node2,node3. . .)
```

/OUTPUT=filename
specifies the name of a file to which you want to direct the output of the SELECT/SHOW PROCESS command. The output is formatted in tabular form.

filename is the name of a file based on OpenVMS file naming conventions.

/PASSIVE_CONNECT=name|*|(list)
is a passive connect record.

name is a 1–16 character name of the passive connect record.

Asterisk (*) is a wildcard character used to indicate the names of all passive connect records.

You can specify a list of passive connect records. The list is enclosed in parentheses, and each node name is separated by a comma; for example:

```
/PASSIVE_CONNECT=(name1,name2,name3. . .)
```

/PRINT
routes the output of the SELECT/SHOW PROCESS command to the default printer. The output will be formatted in tabular form as it would be displayed interactively.

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

Examples

The following examples illustrate the use of the SHOW NETMAP command in noninteractive mode.

The following command displays the network map parameters for the adjacent node named SC.MVS.USER1. Notice that *adjacent* is omitted when selecting the network map for an adjacent_node.

```
$ NDMUI SHOW NETMAP /NODE=SC.MVS.USER1
```

Output from the preceding command is as follows:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          SELECT NETWORK MAP
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Adjacent Node     =>  SC.MVS.USER1     Parsess Max  =>   4
Default Conn      =>  SCAN             Parsess Dep  =>   3

Communications Paths
  NAME=SNAPTH TYPE=SNA GATEWAY=NDM ACCESS.NAME=TEST APPLID=NMVSD38
  NAME=DECPTH TYPE=DECNET DECNET.NODE=NDM DECNET.OBJ=TEK_SRVR
```

The following command displays the NETMAP parameters for the local node from a network map. Notice that the name of the local_node is omitted when selecting the network map for a local_node.

```
$ NDMUI SHOW NETMAP /LOCAL_NODE
```

Output from the preceding command is as follows:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          SELECT NETWORK MAP
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Local Node => MY.NODE
```

The following command displays the NETMAP parameters for all of the passive connects from a network map.

```
$ NDMUI SHOW NETMAP /PASSIVE_CONNECT=*
```

Output from the preceding command is as follows:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          SELECT NETWORK MAP
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

Passive Conn      =>  REC1             Perm Sess    =>   1
Gateway Name      =>  NDM              Access Name  =>   TEST
Circuit Name      =>  LC-1             Session Id   =>   (54-61)
```

Monitoring Processes in the TCQ

The SHOW PROCESS command allows you to monitor all Processes on the TCQ. Issuing the SHOW PROCESS command furnishes you with the Process name and number, submitter node and id, destination node, and the queue, based on the selection criteria specified.

Note: You can substitute SELECT PROCESS for SHOW PROCESS. Output from all SELECT and SHOW commands can be terminated with **CTRL-C**.

Reviewing the Command Format

The following shows the command format and qualifiers for the SHOW PROCESS command. A description of each qualifier follows the SHOW PROCESS command format. There are no parameters.

Command	Qualifiers
SHOW PROCESS	/[NO]CASE
	/DEST=destination_nodename|(list)
	/DETAIL
	/LAST
	/OUTPUT=filename
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/PRINT
	/QUEUE=All|B|E|H|R|S|W|Y|(list)
	/SERVER=server_alias
	/SUBMITTER=("submitter_node,submitter_id")|(list)

Parameters

There are no parameters associated with the SHOW PROCESS command.

Required Qualifiers

You do not have to specify a qualifier with the SHOW PROCESS command. However, if you do not specify a qualifier, all Processes are selected.

Qualifiers

/[NO]CASE
allows for the case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/DEST=destination_nodename|(list)
searches for the Process by destination node, as well as other selection criteria as you choose to specify. Destination nodename is used interchangeably with SNODE (secondary node). This qualifier is especially useful when you need to monitor a Process but several have the same PNAME with different SNODEs.

destination_nodename is the name of the node with which you are communicating. Refer to the network map of your system for node names.

You can specify a list of destination node names. The list is enclosed in parentheses, and each destination node name is separated by a comma.

/DETAIL
shows additional details, such as; submitter class, priority, scheduled start time, scheduled start day and/or date, Process file name, and the value associated with the /RETAIN qualifier. If /RETAIN was not specified, a value is not displayed.

/LAST
selects records for the last Process submitted.

/OUTPUT=filename
specifies the name of a file to which you want to direct the output of the SELECT/SHOW PROCESS command. The output will be formatted in tabular form.

filename is the name of a file based on OpenVMS file naming conventions.

/PNAME=name|(list)
searches for the Process by Process name.

name specifies the name of a specific Process. Process names can range from 1–8 alphanumeric characters.

Asterisk (*) lists all Processes in the TCQ.

Note: The * is valid as a wildcard character with the /PNAME qualifier in the SELECT/SHOW PROCESS command.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER=number|(list)
searches for the Process by Process number.

number specifies the specific Process number of a Process. Process numbers are assigned by Sterling Connect:Direct when Processes are submitted successfully. Process numbers can range from 1–99999.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/PRINT
routes the output of the SELECT/SHOW PROCESS command to the default printer. The output is formatted in tabular form.

/QUEUE=A|B|E|H|R|S|W|Y|(list)
searches for a Process on the specified queue. Possible values are A, B, E, H, R, S, W, and Y. The default is **A**, which stands for All.

The following table defines each of the /QUEUE values.

Queue	Definition
A (All)	All Processes in the TCQ will be displayed. All is the default.
B (Bad)	An error occurred during initiation of Process execution. This can occur because of a security error of some other unrecoverable error.
E (Execute)	The Process is currently executing.
H (Hold)	The Process will remain on the Hold queue until an operator releases it by specifying CHANGE PROCESS where /PNAME=filename and /HOLD=NO. /RELEASE can be used in place of /HOLD=NO.
R (Retain)	The Process will be retained after execution. A copy of this Process can be released for execution by specifying CHANGE PROCESS where /PNAME=filename and /RELEASE.
S (Suspend)	The Process is suspended by the SUSPEND PROCESS command. The Process can be released from the Suspend queue by specifying CHANGE PROCESS where /PNAME=filename and /RELEASE.
W (Wait)	The Process is waiting for execution.
Y (Retry)	The Process is retried after a certain interval if the error that occurred is recoverable.

You can specify a list of queues. The list is enclosed in parentheses, and each queue is separated by a comma; for example:

```
/QUEUE=(A,E,W. . .)
```

/SERVER= server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

/SUBMITTER=(“submitter_node, submitter_id”)|(list)
searches for the Process by the submitter node and submitter id of the Process submitter.

submitter_node is a 1–16 alphanumeric character name that specifies the adjacent node name of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the Process submitter.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

Examples

The following examples illustrate the use of the SHOW PROCESS command in noninteractive mode.

The following command selects the Process named TEST1 with a destination node of SC.VMS.VXNED1:

```
$ NDMUI SHOW PROCESS /PNAME=TEST1 /DEST=SC.VMS.VXNED1
```

The following command selects all Processes:

```
$ NDMUI SHOW PROCESS /PNAME=*
```

The following command selects all Processes and prints the results in tabular form on the default printer:

```
$ NDMUI SHOW PROCESS /PNAME=*/PRINT
```

The following examples show samples of the displays available for monitoring the progress of your transmission.

The following is an example of the SHOW PROCESS display:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
 PNAME    PNUMBER   SUBMITTER NODE   SUBMITTER ID   OTHER NODE   QUEUE
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
 TODALLAS    1      SC.VMS.BOSTON     BOSTON11      SC.VMS.DAL   Exec
```

The following example is a detailed SHOW PROCESS display output:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                          SELECT PROCESS
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Process Name=>  TODALLAS   Submitter => SC.VMS.BOSTON  BOST11
Process Number  => 1       Pnode     => SC.VMS.DAL     Queue  =>  Exec
Submitter Class => NON     Priority  => 10             Retain =>
Schedule Time   =>         Date/Day  =>
Process File    => STEP
Execution Class => 3       State     => Exec Prc+PC\File Write
Executing SM    =>
Step Name       => STEP01  Function  => COPY           Exec Node =>
RECEIVING
Sent:Bytes   => 0      Recs => 0    RUs => 0       Xmit Bytes => 0
Rcvd:Bytes   => 23440  Recs => 293  RUs => 12      Xmit Bytes => 24576
FROM FILE    => SCQA1.FDATA
TO FILE      => NDM$$OUTPUT:MVS_TES1.OUT
```

Displaying Current Server Settings

The SHOW SERVER command shows you the current server setting.

Understanding the Command Format

The following shows the format of the SHOW SERVER command. There are no parameters or qualifiers.

Command
SHOW SERVER

Parameters

There are no parameters associated with the SHOW SERVER command.

Qualifiers

There are no qualifiers associated with the SHOW SERVER command.

Examining Process Statistics

The SHOW STATISTICS command allows you to examine statistics for Processes. The type of information in the detailed statistics report includes such pertinent data as date, Process name and number, PNODE, SNODE, return code, message id, feedback, file name, short message text, the function (subroutine) issuing the message, and the OpenVMS Process name of the session manager issuing the Process.

Examples of the command and a sample statistics log are provided beginning on page 4-56.

The qualifiers used with the command allow you to determine search criteria and the form in which the information is presented. Unless otherwise specified, the output is displayed. You can direct the output to an OpenVMS file or to the default printer.

Note: You can substitute SELECT STATISTICS for SHOW STATISTICS. Output from all SELECT/SHOW commands can be terminated with **CTRL-C**.

Reviewing the Command Format

The following shows the command format and qualifiers for the SHOW STATISTICS command. A description of each qualifier follows the SHOW STATISTICS command format. There are no parameters.

Command	Qualifiers
SHOW STATISTICS	/[NO]CASE
	†/AFTER
	†/BEFORE
	/CCODE=(condition,completion code)
	/DETAIL
	/EXCLUDE=(MEMBER|MCR|WTO|NOTWTO)
	/LAST
	/OUTPUT=filename
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/PRINT
	/SERVER=server_alias
	†/SINCE
	/STARTT=([date|day][,hh:mm:ssXM])
	/STOPT=([date|day][,hh:mm:ssXM])
	/SUBMITTER=("submitter_node,submitter_id")|(list)

† Specify these time-based qualifiers in the same format as other standard OpenVMS commands.

Parameters

There are no parameters associated with the SHOW STATISTICS command.

Required Qualifiers

There are no required qualifiers for the SHOW STATISTICS command.

Qualifiers

/[NO]CASE
allows for the case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/CCODE=(condition,completion code)
searches for records by completion codes for each step of a Process.

condition allows you to set limits on the record search by completion code. The options for specifying a condition are: GT (greater than), LT (less than), EQ (equal), NE (not equal), GE (greater than or equal), or LE (less than or equal).

completion code specifies a completion code value. For example, if CCODE = (GT,0) is specified, you will see statistics records in which the step completion code is greater than zero, as long as the records also meet other specified criteria. A zero completion code indicates that the step was successful.

The standard completion codes are as follows:

- **0** indicates successful execution of the Process step.
- **4** indicates a warning level error was encountered. The command probably completed normally but you should verify the execution results.
- **8** indicates an error occurred during Process step execution.
- **16** indicates a severe error occurred during Process step execution.

/DETAIL
shows additional details such as: submitter class, priority, scheduled start time, scheduled start day and date, Process name, return code, message id, and feedback.

/EXCLUDE=(MEMBER|MCR|WTO|NOTWTO)
specifies that certain statistics records are not selected. If this qualifier is not entered, all statistics are selected.

MEMBER or **MCR** specifies that copy steps involving library modules are excluded.

WTO specifies that informational messages are excluded from the SHOW STATISTICS command.

NOTWTO specifies that all normal statistics are excluded.

/LAST
show statistics records for the last Process submitted. It cannot be specified if the /STARTT qualifier is specified.

/OUTPUT=filename
directs the output of the SHOW STATISTICS command to the specified filename or device. The output is formatted in tabular form, that is, as it is typically displayed online.

filename specifies a filename that conforms to standard OpenVMS naming conventions.

/PNAME=name|(list)
searches for the Process by Process name.

name specifies the name of a specific Process. Process names can range from 1–8 alphanumeric characters.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER=number|(list)
searches for Processes by Process number.

number specifies the specific Process number of a Process. Process numbers are assigned by Sterling Connect:Direct when Processes are successfully submitted. Process numbers can range from 1–99999.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/PRINT
routes the output of the SHOW STATISTICS command to the default printer. The output is formatted in tabular form as it would be displayed online.

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

/STARTT=([date|day][,hh:mm:ssXM])
searches for statistics by start time and either the day or date. Note that *date*, *day*, and *time* are positional parameters. If the date or day is not specified, a null string and a comma must precede the time; for example:

```
/STARTT=("",8:00:00AM)
```

date indicates the beginning date for the records search. You can specify dates in either Gregorian or Julian format.

Gregorian dates always include numeric values for the month (m), day (d), and year (y). Sterling Connect:Direct for OpenVMS only accepts the month-day-year format for Gregorian dates.

Enter Gregorian dates with either a two-digit or a four-digit year. You can use periods and backslashes (/) to separate the values.

Note: You must use a separator to specify a four-digit Gregorian year.

- mmddyy
- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

You can use periods and backslashes (/) to separate the values. Sterling Connect:Direct for OpenVMS processes the following Julian date formats:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only the date is specified, the time will default to 00:00.

Note: If you are using a date format with slashes, start values should be enclosed within double quotation marks; for example:

/STARTT=("09/21/1998")

---

day indicates the beginning day for the record search. If only the day is specified, the time defaults to 00:00. You can also specify **TODAY**.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process executed. XM can be set to AM or PM. You do not have to specify minutes and seconds.

You can express the time using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00 through 24:00. If you use the 12-hour clock, you would express time using AM or PM; therefore, the 24-hour clock is assumed if you do not use AM or PM. You can also specify **NOON** or **MIDNIGHT**.

If the STARTT qualifier is not specified, the search criterion consists of any time before the STOPT qualifier.

If the /LAST qualifier is specified, you cannot specify STARTT.

/STOPT= [date|day][,hh:mm:ssXM])

specifies that statistics records are searched for up to and including the designated *date*, *day*, and *time*. Date, day, and time are positional parameters. If the date or day is not specified, a null string and a comma must precede the time; for example:

```
/STOPT=("",8:00:00AM)
```

date indicates the beginning date for the records search. You can specify dates in either Gregorian or Julian format.

Gregorian dates always include numeric values for the month (m), day (d), and year (y). Sterling Connect:Direct for OpenVMS only accepts the month-day-year format for Gregorian dates.

Enter Gregorian dates with either a two-digit or a four-digit year. You can use periods and backslashes (/) to separate the values.

Note: You must use a separator to specify a four-digit Gregorian year.

- mmddyy
- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

You can use periods and backslashes (/) to separate the values. Sterling Connect:Direct for OpenVMS processes the following Julian date formats:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only the date is specified, the time will default to 00:00.

Note: If you are using a date format with slashes, stop values should be enclosed within double quotation marks; for example:

/STOPT=(“09/21/1998”)

---

day indicates the ending day for the records search. You can also specify **TODAY**.

hh:mm:ssXM indicates the time of day in hours, minutes, and seconds that the last statistics record is selected. XM can be set to AM or PM.

You can express the time using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00 through 24:00. If you use the 12-hour clock, you would express time using AM or PM; therefore, the 24-hour clock is assumed if you do not use AM or PM. Minutes and seconds need not be specified. You also can specify **NOON** or **MIDNIGHT**.

The default for the time is 24:00:00, the end of the day. If the STOPT qualifier is not specified, statistics are selected until the present.

/SUBMITTER=(“submitter_node, submitter_id”)|(list)
searches for statistics of a Process by the submitter node and submitter id of the Process submitter.

submitter_node is a 1–16 alphanumeric character name that specifies the symbolic nodename of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the Process submitter.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

Examples

The following examples illustrate the use of the SHOW STATISTICS command in noninteractive mode.

The following command selects all statistics records for the Process named TEST1 that executed today:

```
$ NDMUI SHOW STATISTICS /PNAME=TEST1 -
_> /STARTT=TODAY
```

The following command selects all statistics records for Processes that executed between 8:00 a.m. and 10:00 p.m. today:

```
$ NDMUI SHOW STATISTICS /STARTT=("",8:00) -
_> /STOPT=("",10:00PM)
```

The following command selects Processes with the criteria: completion codes greater than **0,** Process number equal to **1**, executed since Wednesday at 1:00 p.m.:

```
$ NDMUI SHOW STATISTICS /CCODE=(GT,0) -
_> /PNUMBER=1 /STARTT=(WEDNESDAY,13:00)
```

The following is an example of the SHOW STATISTICS display:

```
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
  DATE          TIME       PNUM     PNAME     EVENT     DESCRIPTION
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
  09.15.1998    08:06:09   1        COPYVMS   SUBMIT    SC.VMS.DAL
```

The following Process generates the statistics records shown in the figure on page 4-57.

```
*THE FOLLOWING PROCESS WILL COPY A SEQUENTIAL FILE,
*SCQA1.FDATA, FROM MVS TO VMS
STEPTEST PROCES              SNODE=SC.MVS.BOSTON
STEP01   COPY      FROM      DSN=SCQA1.FDATA SNODE                          -
                             DISP=SHR)                                      -
                   COMPRESS=PRIME=X'20'                                     -
                   TO        (DSN=U1:<DLSS.TEST>MVS.OUT                     -
                             PNODE DISP=RPL)
```

If the above Process was submitted today and assigned Process number 5, then the following command will select all its statistics records.

```
$ NDMUI SHOW STATISTICS /DETAIL /PNUMBER=5 /STARTT=TODAY
```

The following is the output from the previous command:

```
1.
 = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
                                SELECT STATISTICS
 = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Date       =>    01.24.1999     Time        =>    16:27:51.18        PROCESS-SUBMIT        1
Pnumber    =>    1              Node        =>    SC.MVS.BOSTON
Pname      =>    STEPTEST       Submitter   =>    SC.VMS.DAL     DLSS
Rtncd      =>    0              Message ID  =>    VSRV101I       Feedback   =>  0
File       STEP
VSRV101I:    Feedback:  0         Reply:  0      Function:  dtf_submit
Process submitted successfully. Process number:  1
  ..........................................................................................
Date       =>    01.24.1999     Time        =>    16:28:19.92        PROCESS-PROCSTART     2
Pnumber    =>    1              Snode       =>    SC.MVS.BOSTON  Xnode      =>  P
Pname      =>    STEPTEST       Submitter   =>    SC.VMS.DAL     DLSS
SM Name    =>    QA4_AT_02G                                      Class      =>  4
  ..........................................................................................
Date       =>    01.24.1999     Time        =>    16:28:29.50        PROCESS-STEPSTART     3
Pnumber    =>    1              Snode       =>    SC.MVS.BOSTON  Xnode      =>  P
Pname      =>    STEP_TEST      Submitter   =>    SC.VMS.DAL     DLSS
Function   =>    COPY           Step Name   =>    STEP01
  ..........................................................................................
Date       =>    01.24.1999     Time        =>    16:30:55.45        PROCESS-STEPEND       4
Pnumber    =>    1              Xlate       =>                   Start Date =>  01.24.1999
Pname      =>    STEPTEST       Compress    =>    YES            Start Time =>  16:27:51.18
Msgid      =>    SCPA000I       Restart     =>    NO             End Date   =>  01.24.1999
Rtncd      =>    0              Link Stat   =>    OK             End Time   =>  16:30:48.21
FDBK       =>    0              Snode       =>    SC.MVS.BOSTON  Direction  =>  RECEIVE
Step       =>    STEP01         Submitter   =>    SC.VMS.DAL     DLSS
           From  Snode  DSN=  SCQA1.FDATA                                RUsize=   2048
                 I/O Bytes=>  80000              Xmit Bytes=>  78483
                 I/O Recs =>  1000               Xmit RUs=>    39      Comp%=> 1.90
           To    Pnode  DSN= U1:<DLSS.TEST>:MVS.OUT
                 I/O Bytes=>  80000              Xmit Bytes=>  78483
                 I/O Recs=>   1000               Xmit RUs=>    39      Comp%=> 1.90
SCPA000I:    Feedback:  0         Reply:  0      Function:  copy_function
COPY successfully completed.
  ..........................................................................................
Date       =>    01.24.1999     Time        =>    16:30:58.98        MESSAGE-MSG           5
Pnumber    =>    1              Node        =>    SC.MVS.BOSTON  SM Name    =>  QA4_AT_02G
Pname      =>    STEPTEST       Submitter   =>    SC.VMS.DAL     DLSS
Exit step, ending process execution - (execute_process)
  ..........................................................................................
Date       =>    01.24.1999     Time        =>    16:31:04.40        PROCEND               6
Pnumber    =>    1              Snode       =>    SC.MVS.BOSTON  Xnode      =>  P
Pname      =>    STEPTEST       Submitter   =>    SC.VMS.DAL     DLSS
Rtncd      =>    0              Message ID  =>    VSMT000I       Feedback   =>  0
VSMT000I:    Feedback:  0         Reply:  0      Function:  execute_process
End of Process.
```

The various elements of the statistics log shown in the previous figure are divided into six records: SUBMIT, PROCSTART, STEPSTART, STEPEND, MSG, and PROCEND. Each section in the previous figure is labeled. A description of each line in the records is provided.

1 PROCESS-SUBMIT

PROCESS-SUBMIT provides statistics about the submittal of the Process.

Date
indicates the date the Process was submitted to the TCQ.

Time
indicates the time the Process was submitted to the TCQ.

PROCESS-SUBMIT
is the label that describes this phase of the submitted Process.

Pnumber
is the Process number that was assigned by Sterling Connect:Direct when the Process was submitted successfully.

Node
is the secondary nodename.

Pname
is the label on the PROCESS statement.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's OpenVMS user name.

Rtncd
is the completion code for the step. A zero (0) indicates successful completion of a step. The field can be shown in decimal or hexadecimal format.

Message ID
is the identification number of the message in the online message file.

Feedback
is optional information for diagnostic purposes.

File
is the name of the file containing the Process that you are submitting.

VSRV101I
is the identification number of the message in the online message file. A short version follows the message id number. In this case, the *Process submitted successfully.*

Feedback
is repeated as part of the short message text and provides optional information for diagnostic purposes.

Reply
is the return code that is repeated as part of the short message. It is the completion code for the step. A zero (**0**) translates to successful completion of a step.

Function
is the subroutine name for the particular step in the Process. It is a name that is part of Sterling Connect:Direct internals and is used for diagnostic purposes.

Process submitted successfully
is the short version of the message text. If you need further information, refer to the online message file, using the message id number that is provided as part of this record.

Process number
is repeated as part of the short message text and is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

2 PROCESS-PROCSTART

PROCESS-PROCSTART provides statistics about the start of the Process.

Date
indicates the date the Process actually started running.

Time
indicates the time the Process started running.

PROCESS-PROCSTART
is the label that describes this phase of the submitted Process.

Pnumber
is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

Snode
is the secondary nodename.

Xnode
indicates the node with Process control. In this example, the Xnode is the PNODE.

Pname
is the label on the PROCESS statement.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's OpenVMS user name.

SM Name
is the OpenVMS process name of the session manager that is executing the Process.

Class
defines the session used.

3 PROCESS-STEPSTART

PROCESS-STEPNAME provides statistics about the start of a step of a Process.

Date
indicates the date the particular step of a Process actually started running.

Time
indicates the time the particular step of a Process started running.

PROCESS-STEPSTART
is the label that describes this phase of the submitted Process.

Pnumber
is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

Snode
is the secondary nodename.

Xnode
indicates the node that is transmitting the data during this step. In this example, the Xnode is the PNODE.

Pname
is the label on the PROCESS statement.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's OpenVMS user name.

Function
is the subroutine name for the particular step in the Process. It is a name that is part of Sterling Connect:Direct internals and is used for diagnostic purposes. In this case, it identifies the COPY statement entered as part of the Process.

Step Name
is the label in the Process.

4 PROCESS-STEPEND

PROCESS-STEPEND provides statistics about the end of a step of a Process.

Date
indicates the date the particular step of a Process stopped running.

Time
indicates the time the particular step of a Process stopped running.

PROCESS-STEPEND
is the label that describes this phase of the submitted Process.

Pnumber
is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

Xlate
indicates that you are converting from EBCDIC to ASCII.

Start Date
indicates the date the Process actually started running.

Pname
is the label on the PROCESS statement.

Compress
indicates whether you have requested data compression of the data that was transmitted.

Start Time
indicates the time the Process started running.

Message ID
is the identification number of the message in the online message file.

Restart
indicates whether the transmission was interrupted and the Process was restarted.

End Date
is the ending date for the particular step in the Process.

Rtncd
is the completion code for the step. A zero (0) translates to successful completion of a step. The field can be shown in decimal or hexadecimal format.

Link Stat
indicates the status of the communications link to the node to which you are transmitting. Values for this field include OK and failed.

End Time
is the ending time for the particular step in the Process.

FDBK (feedback)
is optional information for diagnostic purposes.

Snode
is the secondary nodename.

Direction
is the direction of the files transfer. Values for this field are SEND and RECEIVE.

Step
is the label name as assigned by the user as part of the Process.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's user id.

From Snode DSN
is the name of the data set from which the data is being transmitted. In this example, the Process is submitted from the SNODE.

To Pnode DSN
is the name of the data set that is receiving the transmission. In this example, the Process is received at the PNODE.

I/O Bytes
fields indicate the number of bytes that are read or written from disk or tape. This information is provided for both SNODE and PNODE.

Xmit Bytes
fields indicate the number of bytes sent or received during the session, including Sterling Connect:Direct control information. This information is provided for both SNODE and PNODE.

Comp%
indicates the compression percentage. This information is provided for both SNODE and PNODE.

I/O Recs
indicate the number of actual records transmitted. This information is provided for both SNODE and PNODE.

Xmit RUs
indicate the number of request units transmitted. This information is provided for both SNODE and PNODE.

SCPA000I
is the identification number of the message in the online message file. A short version follows the message id number. In this example, the *COPY successfully completed*.

Feedback
is repeated as part of the short message text and provides optional information for diagnostic purposes.

Reply
is the return code that is repeated as part of the short message; it is the completion code for the step. A zero (0) translates to successful completion of a step.

Function
is the subroutine name for the particular step in the Process; it is a name that is part of Sterling Connect:Direct internals and is used for diagnostic purposes.

COPY successfully completed
is the short version of the message text. If you need further information, refer to the online message file, using the message id number that is provided as part of this record.

5 MESSAGE-MSG

MESSAGE-MSG provides statistics about informational messages.

Date
indicates the date of the message.

Time
indicates the time of the message.

MESSAGE-MSG
is the label that describes this phase of the submitted Process.

Pnumber
is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

Node
is the secondary nodename.

SM Name
is the OpenVMS process name of the session manager that is executing the Process.

Pname
is the label on the PROCESS statement.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's OpenVMS user name.

Exit step, ending process execution - (execute_process)
is the message indicating that execution is complete.

6 PROCEND

PROCEND provides statistics about the end of the Process.

Date
indicates the date the Process ended.

Time
indicates the time the Process ended.

PROCEND
is the label that describes this phase of the submitted Process.

Pnumber
is the number assigned by Sterling Connect:Direct when the Process was submitted successfully.

Snode
is the secondary nodename.

Xnode
indicates the node that is transmitting the data during this step. In this case, the Xnode is the PNODE.

Pname
is the label on the PROCESS statement.

Submitter
is the name of the node submitting the Process. DLSS is the submitter's OpenVMS user name.

RTNCD
is the completion code for the step. A zero (0) translates to successful completion of a step. The field can be shown in decimal or hexadecimal format.

Message ID
is the identification number of the message in the online message file.

Feedback
is optional information for diagnostic purposes.

VSTM000I
is the identification number of the message in the online message file. A short version follows the message id number.

Feedback
is repeated as part of the short message text and provides optional information for diagnostic purposes.

Reply
is the return code that is repeated as part of the short message; it is the completion code for the step. A zero (0) translates to successful completion of a step.

Function
is the subroutine name for the particular step in the Process; it is a name that is part of Sterling Connect:Direct internals and is used for diagnostic purposes.

End of Process
is the short version of the message text. If you need additional information, refer to the online message file, using the message id number that is provided as part of this record.

Obtaining Current Version

The SHOW VERSION command allows you to display the current version of Sterling Connect:Direct for OpenVMS.

Reviewing the Command Format

The following shows the format of the SHOW VERSION command. There are no parameters or qualifiers.

Command	Parameters
SHOW VERSION	

Parameters

There are no parameters associated with the SHOW VERSION command.

Qualifiers

There are no qualifiers associated with the SHOW VERSION command.

Example

Results of the SHOW VERSION command are displayed as follows:

```
Sterling Connect:Direct for OpenVMS Version 3.6.00
```

Issuing DCL Commands

The SPAWN command is useful when you are in *interactive mode***.** It creates a subprocess that allows you to issue DCL commands at the DCL prompt without terminating your Sterling Connect:Direct session.

SPAWN can work in two ways. If you issue the SPAWN command without specifying a DCL command, you can work at the DCL prompt for an extended period of time. You can return to interactive mode at any time by entering the DCL LOGOUT command, or you can issue the SPAWN command with a single DCL command. Once the DCL command executes, you return to interactive mode without entering LOGOUT.

Reviewing the Command Format

The following shows the command format of the SPAWN command.

Command
SPAWN

Stopping Sterling Connect:Direct

The STOPNDM command allows you to stop Sterling Connect:Direct while it is executing. If you are issuing commands in interactive mode and issue the STOPNDM command, you also need to exit the NDMUI by issuing the EXIT command.

Reviewing the Command Format

The following shows the command format and qualifiers of the STOPNDM command. A description of the qualifiers follows the STOPNDM command format. Default values are underlined. There are no parameters.

Command	Qualifiers
STOPNDM	/FORCE

Parameters

There are no parameters associated with the STOPNDM command.

Qualifiers

There are no required qualifiers for the STOPNDM command

/FORCE
stops Sterling Connect:Direct immediately; the server immediately cancels all session managers.

Examples

The following examples illustrate the STOPNDM command in noninteractive and interactive modes.

In noninteractive mode, STOPNDM stops Sterling Connect:Direct upon completion of the executing Processes. The prompt returns to the OpenVMS prompt.

```
$ NDMUI STOPNDM
```

In interactive mode, STOPNDM stops Sterling Connect:Direct upon completion of the executing Processes. The prompt returns to user interface after the STOPNDM command is executed. The EXIT command is required in order to return to the OpenVMS prompt.

```
$ NDMUI
CONNECT:Direct> STOPNDM
CONNECT:Direct> EXIT
$
```

Submitting a Process

Use the SUBMIT command to place a Process in the TCQ for execution.

Reviewing the Command Format

The following shows the format, parameters, and qualifiers of the SUBMIT command. The required parameter is in bold print. Qualifiers are indicated with a forward slash character (/). A description of the parameter and qualifiers follows the SUBMIT command format. Default values for qualifiers are underlined.

Command	Parameters and Qualifiers
SUBMIT	filename
	†/AFTER
	/[NO]CASE
	/CLASS=number
	/DEFCONN_MODE=FIRST|SCAN|name
	/HOLD=No|Yes
	/LOG[=file]
	/MAIL=[username|(list)]
	/NOKEEP
	/NORESTART
	/NOTIFY=[username|(list)]
	/PNAME=name
	/PRTY=number
	/RETAIN=No|Yes|Initial
	/RETRY_LIMIT=number
	/SACCT='snode-accounting-data'
	/SERVER=server_alias
	/SNODE=nodename
	/SNODEID=(id, pswd [,newpswd])
	/STARTT=([date|day] [,hh:mm:ssXM])
	/SYMBOLICS=(symbol=value [,symbol1=value1])
	/TEST
	/[NO]WAIT
	/XSID=REMOTE_ACCT_IDENTIFIER

† Specify this time-based qualifier in the same format as other standard OpenVMS commands.

Required Parameters

filename

names the file you are submitting that contains the Process. The filename can be any valid OpenVMS filename. An extension of NDM is not required; however NDM is assumed if an extension is not provided.

Qualifiers

There are no required qualifiers for the SUBMIT command.

/[NO]CASE
allows for the case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/CLASS=number
determines the node-to-node session on which a Process can execute. If CLASS is not specified in your Process, it will default to the class value specified for PARSESS in the network map.

/DEFCONN_MODE=FIRST|SCAN|name
specifies the default method of selecting a communication path that is used to establish a session for Process execution.

FIRST specifies that Sterling Connect:Direct uses only the first COMM_PATH specification to establish a connection.

SCAN specifies that Sterling Connect:Direct uses each COMM_PATH specification in turn until a connection is successfully made.

name specifies an actual COMM_PATH name which Sterling Connect:Direct uses when establishing a connection.

/HOLD=No|Yes
delays execution of a Process until it is released or until it is *called* by a remote node.

If you specify both HOLD=YES and a STARTT (start time and/or start date) value, the Process is placed in the Hold queue. If the start time has not passed once you release the Process, the Process begins execution at the specified start time.

If you specify RETAIN=YES and HOLD=NO, HOLD is ignored.

No specifies that the Process is executed immediately. The default is No.

Yes specifies that the Process remains in the Hold queue until you take one of the following actions:

- Release the Process with the CHANGE PROCESS /HOLD=NO command.
- Delete the Process with the DELETE PROCESS command.
- Release the Process with the CHANGE PROCESS /RELEASE command.

/LOG[=file]
causes Process termination status messages to be written to a file.

file is the user-defined specific log file. The default file specification is SYS$LOGIN:NDM_EVENT.LOG.

/MAIL=[username|(list)
causes Process termination status messages to be mailed to the specified user name or list of user names.

username is the user name to receive the status message. The default user name is the command issuer.

You can specify a list of user names. The list is enclosed in parentheses, and each user name is separated by a comma.

```
/MAIL=(username1,username2 ,username3. . .)
```

/NOKEEP
indicates that the Process is not to be retained in the Hold queue when it has exhausted its retry limit.

/NORESTART
indicates that the Process cannot be restarted in case of node or server failure.

/NOTIFY=[username|(list)
causes Process termination status messages to be broadcast to a user name or list of user names.

username is the user name to receive the status broadcast. The default user name is the command issuer.

You can specify a list of user names. The list is enclosed in parentheses, and each user name is separated by a comma.

```
/NOTIFY=(username1, username2. . .)
```

/PNAME=name
overrides the label on the PROCESS statement in the statistics records for this operation.

name is the 1–8 character alphanumeric name of the Process.

/PRTY=number
sets the priority of the Process in the TCQ. This priority is used for Process selection and does not affect the OpenVMS priority. The default priority is defined during installation of Sterling Connect:Direct.

number is an integer in the range 0–15.

/RETAIN=No|Yes|Initial
keeps a copy of the Process in the TCQ after it executed and establishes other times and days that it will execute. The Process number of the copy is incremented by 100,000.

If you enter the SUBMIT command with /RETAIN=YES, a date cannot be used with /STARTT. You can only specify a start time and day or start time or day.

If you enter the SUBMIT command with /RETAIN=YES and a STARTT ***(start time only)*** value, the Process executes at the same time every day.

If you enter the SUBMIT command with /RETAIN=YES and STARTT ***(start time and day)*** values, a Process executes at the specified time and day each week.

If you enter the SUBMIT command with only /RETAIN=YES, the Process is placed on the Hold queue and can be released with the CHANGE PROCESS /RELEASE command.

The /STARTT qualifier is not valid with /RETAIN=INITIAL.

No deletes the Process after execution. The default is No.

Yes keeps a copy of the Process in the TCQ after execution and executes it as instructed by the STARTT values.

Initial keeps a copy of the Process and executes it every time that Sterling Connect:Direct is initialized.

/RETRY_LIMIT=number
is used to override the default server table value.

number is an integer indicating the number of retries.

/SACCT='snode-accounting-data'
specifies the accounting data for the secondary node.

snode-accounting-data has a maximum length of 256 characters. If special characters or blanks are part of the accounting data, the string must be enclosed in double quotation marks.

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

/SNODE=nodename
overrides the default SNODE (secondary node) assigned in the Process on the PROCESS statement.

nodename is the alphanumeric character name of the secondary node. It can contain up to 16 characters.

/SNODEID=(id, pswd [, newpswd])
specifies security userids and security passwords at the SNODE.

id specifies the security id that will be passed to a security exit.

pswd specifies the current security password. This parameter can be used by the security exit to validate the current security password.

newpswd specifies the new security password. This changes the current security password to the new security password.

/STARTT=([date|day][,hh:mm:ssXM])
specifies the start time and either the date or day the Process is to execute. Note that *date*, *day*, and *time* are positional; therefore, if the date or day is not specified, a null string and a comma must precede the time. The following example specifies the Process runs on the current date at 8:00 a.m. If 8:00 a.m. is already past, the Process will run immediately.

```
/STARTT=("",8:00:00AM)
```

/STARTT is not valid if you specify /RETAIN=INITIAL.

You cannot specify a date within the /STARTT qualifier with /RETAIN =YES; you can only specify day and time.

date ensures that the Process executes on the specified date. You can specify dates in either Gregorian or Julian format.

Gregorian dates always include numeric values for the month (m), day (d), and year (y). Sterling Connect:Direct for OpenVMS only accepts the month-day-year format for Gregorian dates.

Enter Gregorian dates with either a two-digit or a four-digit year. You can use periods and backslashes (/) to separate the values.

Note: You must use a separator to specify a four-digit Gregorian year.

- mmddyy
- mm/dd/yy *or* mm/dd/yyyy
- mm.dd.yy *or* mm.dd.yyyy

You can use periods and backslashes (/) to separate the values. Sterling Connect:Direct for OpenVMS processes the following Julian date formats:

- yyddd *or* yyyyddd
- yy/ddd *or* yyyy/ddd
- yy.ddd *or* yyyy.ddd

If only the date is specified, the time will default to 00:00.

Note: If you are using a date format with slashes, start values must be enclosed within double quotation marks; for example:

/STARTT=("09/21/1998")

---

day releases the Process for execution on the specified day of the week.

If only the day is specified, the time will default to 00:00; therefore, if you submit a Process on Monday, with Monday as the only /STARTT qualifier, the Process will not run until the following Monday.

You also can specify **TODAY**, which releases the Process for execution today, or **TOMORROW**, which releases the Process for execution the next day.

hh:mm:ssXM indicates the time of day in hours (hh), minutes (mm), and seconds (ss) that the Process is to be released. XM can be set to AM or PM. You do not have to specify minutes and seconds.

You can express the time using the 24-hour clock or the 12-hour clock. If you use the 24-hour clock, valid times are from 00:00 through 24:00. If you use the 12-hour clock, you would express time using AM or PM; therefore, the 24-hour clock is assumed if you do not use AM or PM.

If you specify hh:mm:ssXM and you use /RETAIN=YES, then the Process will execute the same time every day.

You can also specify **NOON**, which releases the Process for execution at noon, or **MIDNIGHT**, releases the Process for execution at midnight.

/SYMBOLICS= (symbol=value [,symbol1=value1])
specifies a set of symbol substitutions in the Process. The substitutions override any default values specified in the PROCESS statement. Values specified in a **symbol** statement cannot be overridden.

A null value can be specified if the equal (=) sign is immediately followed by a comma. Symbolics containing special characters must be enclosed in double quotation marks.

/TEST
allows you to check the syntax and modal logic of a Process without having to actually submit the Process for execution.

/[NO]WAIT
causes the operation to be performed asynchronously. NOWAIT is the default.

/XSID=REMOTE_ACCT_IDENTIFIER
a string identifier that references a specific [username,password] entry in a password file.

Examples

The following example illustrates the SUBMIT command and the /SYMBOLICS qualifier in noninteractive mode.

Assume you have the following Process:

```
SEND   PROCESSNODE=MVS.NODE
STEP01 COPY FROM(DSN=&FROM PNODE)-
            TO   (DSN=&TO DISP=RPL)
```

The following command submits the Process. The values for the TO and FROM data set names specified in the /SYMBOLICS qualifier will be substituted in the Process.

```
$ NDMUI SUBMIT SEND.TXT-
_> /SYMBOLICS=("TO=MVS.FILE","FROM=VMS.TXT")
```

Interrupting an Executing Process

The SUSPEND PROCESS command allows you to interrupt an *executing* Process.

Reviewing the Command Format

The following shows the format and qualifiers of the SUSPEND PROCESS command. Required qualifiers are in bold print. A description of the qualifiers follows the SUSPEND PROCESS command format. There are no parameters.

Command	Qualifiers
SUSPEND PROCESS	/[NO]CASE
	/PNAME=name|(list)
	/PNUMBER=number|(list)
	/SUBMITTER=("submitter_node,submitter_id")|(list)
	/FORCE
	/SERVER=server_alias

Parameters

There are no parameters associated with the SUSPEND PROCESS command.

Required Qualifiers

If **/PNAME**, **/PNUMBER**, or **/SUBMITTER** are not specified, then the command will act on all processes that are accessible by the user. Additional qualifiers are optional.

Qualifiers

/[NO]CASE
allows for the case sensitivity as follows:

- /CASE ensures that case is preserved for all characters enclosed in quotation marks.
- /NO CASE ignores the case of characters enclosed in quotation marks and converts them to uppercase.

/PNAME=name|(list)
searches for the Process by Process name.

name specifies the name of a specific Process. Process names can range from 1–8 alphanumeric characters.

You can specify a list of Process names. The list is enclosed in parentheses, and each Process name is separated by a comma; for example:

```
/PNAME=(PROC1,PROC2,PROC3. . .)
```

/PNUMBER=number|(list)
searches for the Process by Process number.

number specifies the specific number of a Process. These numbers are assigned by Sterling Connect:Direct when Processes are submitted successfully. Process numbers can range from 1–99999.

You can specify a list of Process numbers. The list is enclosed in parentheses, and each Process number is separated by a comma; for example:

```
/PNUMBER=(1,2,3. . .)
```

/SUBMITTER=(“submitter_node, submitter_id”) |(list)
searches for the Process by the submitter node and submitter id of the submitter of the Process.

submitter_node is a 1–16 alphanumeric character name that specifies the symbolic node name of the node.

submitter_id is the alphanumeric character name that specifies the OpenVMS user name of the submitter of the Process.

You can specify a list of submitter nodes and submitter ids. Each submitter_node and submitter_id pair is enclosed in double quotes and is separated by a comma; for example:

```
/SUBMITTER=("submitter_node1,submitter_id1"-
,"submitter_node2,submitter_id2", . . .)
```

/FORCE
suspends the Process by stopping the session manager. You can use this qualifier to terminate the OpenVMS process that is executing the Sterling Connect:Direct script. The Sterling Connect:Direct script is then retained in the TCQ.

/SERVER=server_alias
specifies the particular server to receive the command.

server_alias specifies the 1–16 alphanumeric character name of the server as defined in NDM$$DIRECTORY:SERVER.DAT.

Examples

The following examples illustrate the use of the SUSPEND PROCESS command in noninteractive mode.

The following command suspends an executing Process named TEST1 and places it on the TCQ:

```
$ NDMUI SUSPEND PROCESS /PNAME=TEST1
```

The following command suspends an executing Process number **3** and places it on the TCQ:

```
$ NDMUI SUSPEND PROCESS /PNUMBER=3
```

The following command suspends an executing Process submitted by nodename SC.VMS.QA4 and userid of QA11 and places it in the TCQ:

```
$ NDMUI SUSPEND PROCESS /SUBMITTER=("SC.VMS.QA4,QA11")
```

The following command suspends a hanging Process named TEST1 and places it on the TCQ:

```
$ NDMUI SUSPEND PROCESS /PNAME=TEST1 /FORCE
```

Chapter 5

Using the Application Programming Interface

The Application Programming Interface (API) provides a precise and controlled interface to Sterling Connect:Direct for OpenVMS. The API does replace the user interface program, but rather supplements it. It does not provide all the features and functions of Sterling Connect:Direct for OpenVMS; however, the provided capabilities cover the vast majority of normal situations.

The following compiled script routines are currently supported:

- NDM_CSX_API_SCRIPT_EXEC_SUBMIT
- NDM_CSX_API_SCRIPT_TERM_NOTIFY

The following event API routines are currently supported:

- NDM_EVENT_API_RECEIVE STREAM
- NDM_EVENT_API_DECODE_MESSAGE
- NDM_EVENT_API_DECODE_ITEM
- NDM_EVENT_API_WRITE_MESSAGE
- NDM_EVENT_API_MSGFILE_OPEN
- NDM_EVENT_API_MSGFILE_DISPLAY
- NDM_EVENT_API_MSGFILE_CLOSE
- NDM_EVENT_API_GET_VERSION

This chapter describes the API, its functions and supported routines, and its usage.

Understanding the API

The API follows standard OpenVMS conventions and can be used by any supported OpenVMS language. All routines return standard condition values in register zero (R0) and can be used by any supported language on OpenVMS systems. Because these routines require ast's to complete their processing, ast's should not generally be disabled and ast states are unaffected by these routines. To access these routines, link against the appropriate shareable image.

CSX Application Programming Interface

You can submit compiled scripts under direct program control, by using a shareable image (NDM_CSX_SHR.EXE). It contains routines that obey the OpenVMS procedure calling standards.

The following sections describe the currently defined procedures.

Submitting a Compiled Script for Execution

The NDM_CSX_API_SCRIPT_EXEC_SUBMIT (Submit Compiled Script for Execution) routine performs the following tasks:

- Opens the specified script file
- Performs placeholder substitution
- Issues an execution request to the Sterling Connect:Direct for OpenVMS server process.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_CSX_API_SCRIPT_EXEC_SUBMIT routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_CSX_API_SCRIPT_EXEC_SUBMIT	sx_file
	[sx_env]
	[sx_notify]
	sx_rqid
	rqid_size
	sx_name
	sx_number
	reserved_1
	reserved_2

Return Values

The following are values returned in register zero (0) for the NDM_CSX_API_SCRIPT_EXEC_SUBMIT routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for each argument for the NDM_CSX_API_SCRIPT_EXEC_SUBMIT routine:

sx_file
the filename of the script to be submitted for execution. This is the address of a descriptor pointing to the script filename.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

sx_env
the Sterling Connect:Direct for OpenVMS server process environment name. This is the address of a descriptor pointing to the environment name.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

sx_notify
the address of a longword that contains a notification context identifier created by a previous call to NDM_CSX_API_SCRIPT_TERM_NOTIFY.

OpenVMS Usage:	identifier
type:	longword (unsigned)
access:	read only
mechanism:	by reference

sx_rqid
the address of a descriptor pointing to a character string where NDM_CSX_API_SCRIPT_EXEC_SUBMIT writes the request identifier. The request identifier can be up to 64 bytes in length.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

rqid_size
the length of the request identifier returned by NDM_CSX_API_SCRIPT_EXEC_SUBMIT. The **rqid_size** argument is the address of a unsigned word integer into which NDM_CSX_API_SCRIPT_EXEC_SUBMIT writes the length.

OpenVMS Usage:	word_unsigned
type:	word (unsigned)
access:	write only
mechanism:	by reference

sx_name
the address of a descriptor pointing to a character string into which NDM_CSX_API_SCRIPT_EXEC_SUBMIT writes the name assigned the script by the user. The script name can be up to **8** bytes in length.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

name_size
the length of the script name returned by NDM_CSX_API_SCRIPT_EXEC_SUBMIT. This is the address of a unsigned word integer into which NDM_CSX_API_SCRIPT_EXEC_SUBMIT writes the length.

OpenVMS Usage:	word_unsigned
type:	word (unsigned)
access:	write only
mechanism:	by reference

sx_number
the address of a signed longword into which NDM_CSX_API_SCRIPT_EXEC_SUBMIT writes the number assigned to the execution of the script by Sterling Connect:Direct for OpenVMS.

OpenVMS Usage:	longword_signed
type:	ongword integer (signed)
access:	write only
mechanism:	by reference

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Using the NDM_CSX_API_SCRIPT_EXEC_SUBMIT Routine

NDM_CSX_API_SCRIPT_EXEC_SUBMIT opens the specified compiled script file and performs placeholder substitution and remote access control processing as necessary. The script submit symbols are cleared and the resulting script is issued to the appropriate Sterling Connect:Direct for OpenVMS server process given by the environment name. Upon successful submission, the request identifier assigned to the script execution is returned to the process.

Multiple submission requests can use the returned identifiers to distinguish between outstanding requests for notification purposes.

TMPMBX, NETMBX, OPER privileges are required to use this routine.

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Value Returned	Meaning
SS$_NORMAL	The script was successfully submitted.
SS$_BADPARAM	A reserved argument was not specified or is not zero.
SS$_UNSUPPORTED	The specified file is not recognized as a compile script file.
SS$_BADCHKSUM	A checksum error was encountered in accessing the script file.
SS$_RESULTOVF	An output descriptor specified a buffer which is too small to contain the information to be returned.

Note: Any condition value returned by RMS, $QIO, $CREMBX, or LIB$ANALYZE_SDESC is also stored in register zero (0).

Enabling Script Termination Notification

The NDM_CSX_API_SCRIPT_TERM_NOTIFY (Enable Script Termination Notification) routine enables the Process to receive notification of compiled script termination.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_CSX_API_SCRIPT_TERM_NOTIFY routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_CSX_API_SCRIPT_TERM_NOTIFY	notify_procedure
	[notify_context]
	notify_ident
	reserved_1
	reserved_2

Return Values

The following are values returned in register zero (0) for the NDM_CSX_API_SCRIPT_TERM_NOTIFY routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for each argument for the NDM_CSX_API_SCRIPT_TERM_NOTIFY routine:

notify_procedure
the address of a user-supplied routine that NDM_CSX_API_SCRIPT_TERM_NOTIFY calls when a message indicating script termination is received.

OpenVMS Usage:	procedure
type:	procedure value
access:	function call (before return)
mechanism:	by value

notify_context
the user-supplied argument that NDM_CSX_API_SCRIPT_TERM_NOTIFY passes to the notification procedure. Whatever mechanism is used to pass notify_context to NDM_CSX_API_SCRIPT_TERM_NOTIFY is also used to pass it to the notify routine. If this argument is omitted, a zero value is passed.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

notify_ident
the address of an unsigned longword that receives a notify process identifier. The address of this argument can be specified in subsequent script submissions for which notification of termination is desired.

OpenVMS Usage:	identifier
type:	longword (unsigned)
access:	write only
mechanism:	by reference

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Using the NDM_CSX_API_SCRIPT_TERM_NOTIFY Routine

NDM_CSX_API_SCRIPT_TERM_NOTIFY enables the calling process to receive notification from Sterling Connect:Direct for OpenVMS that the execution of a compiled script has terminated.

The **notify_ident** argument can be specified as a parameter in subsequent calls to NDM_CSX_API_SCRIPT_EXEC_SUBMIT to request notification of script termination as desired. If

notification of script termination is desired, this routine is called prior to any subsequent calls to NDM_CSX_API_SCRIPT_EXEC_SUBMIT.

Call Format for the Notify Routine

The following table shows the calling format and arguments for the notify routine. A description of each argument follows the routine format.

Routine	Arguments
notify_procedure	notify_context
	term_status
	process_id
	process_name
	procesS_user
	sx_rqid
	sx_name
	sx_number
	reserved_1
	reserved_2

Arguments

The following are the arguments for the notify routine:

notify_context
value passed by NDM_CSX_API_SCRIPT_TERM_NOTIFY to the notify routine. The same passing mechanism that was used to pass **notify_context** to NDM_CSX_API_SCRIPT_TERM_NOTIFY is used by NDM_CSX_API_SCRIPT_TERM_NOTIFY to pass **notify_context** to the notify routine.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

term_status
value assigned by the user or by Sterling Connect:Direct for OpenVMS at the termination of the compiled script.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

process_id
the process identifier of the process that submitted the script to be executed.

OpenVMS Usage:	process_identifier
type:	longword (unsigned)
access:	read only
mechanism:	by value

process_name
the process name of the process that submitted the script to be executed. The **process_name** argument is the address of a descriptor that points to a character string containing the process name.

OpenVMS Usage:	process_name
type:	character-coded text string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

process_user
the process user name of the process that submitted the script to be executed. The **process_user** argument is the address of a descriptor that points to a character string containing the process user name.

OpenVMS Usage:	char_string
type:	character-coded text string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

sx_rqid
the request identifier assigned to the script at the time of submission. The **sx_rqid** argument is the address of a descriptor that points to a character string containing the request identifier.

OpenVMS Usage:	char_string
type:	character-coded text string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

sx_name
the name assigned to the script by the user. The **sx_name** argument is the address of a descriptor that points to a character string containing the script name.

OpenVMS Usage:	char_string
type:	character-coded text string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

sx_number
the number assigned to the execution of the script by Sterling Connect:Direct for OpenVMS.

OpenVMS Usage:	user_arg
type:	longword (signed)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Notification was successfully enabled.
SS$_BADPARAM	A reserved argument was not specified or is not zero.

Note: Any condition value returned by $QIO, $CREMBX, LIB$GET_VM, $GETDVI, LIB$GET_EF is also stored in register zero (0).

Event Application Programming Interface

An application programming interface, API, in the form of a shareable image allows you to have complete control over processing event messages.

The API defines a set of global symbols beginning with a prefix that corresponds to a specific function. The symbols and functions are as follows:

Symbol	Function
NDMEVLC_	Specifies event class codes.
NDMEVLT_	Specifies event type codes.
NDMEVLIx_	Symbols with this prefix specify event item type codes of type x as follows: NDMEVLIF_ Logical Flag Value NDMEVLIB_ Unsigned Byte NDMEVLIW_ Unsigned Word NDMEVLIL_ Unsigned Longword NDMEVLIQ_ Unsigned Quadword NDMEVLIT_ OpenVMS Time Quadword NDMEVLIS_ ASCII String NDMEVLIV_ Vector Block

The preceding symbols also define the item type codes.

When linking against the API, you must use the constructs and conventions provided in the language being used to reference global symbols. The following are sample program segments that display the value of the NDMEVLC_EVENT_FACILITY class code:

```
*  Macro
                .external     NDMEVL$C$_EVENT_FACILITY
                .external     display_longword
                ...
                pushl         #NDMEVL$C$_EVENT_FACILITY
                calls         #1,L^display_longword
                ...
                ret
*  C
                globalvalue  int  NDMEVL$C$_EVENT_FACILITY  ;
                display_longword ( NDMEVL$C$_EVENT_FACILITY )  ;
                return
*  Fortan
                EXTERNAL    NDMEVL$C$_EVENT_FACILITY
                DISPLAY_LONGWORD ( %LOC(NDMEVL$C$_EVENT_FACILITY ))
                RETURN
```

The following template equate files are defined in the NDM_SAMPSRC.TLB library:

NDM$EVENT_CLASS.EQU
Class Code Equates

NDM$EVENT_TYPE.EQU
Type Code Equates

NDM$EVENT_ITEM.EQU
Item Code Equates

You can process event messages as they are received or later by reading the event log files.

To process event messages in near real-time, a process calls NDM_EVENT_API_RECEIVE_STREAM. When a message arrives, the caller's action routine is called with the event message class and type to enable the caller to more efficiently filter messages without first having to decode them. In this way, only those messages that pass the caller's event filter need to be decoded. To discard a message, the caller's action routine simply returns.

Alternatively, you can process event messages on a deferred basis by reading the event message log files. After an event log record has been retrieved, the process normally calls NDM_EVENT_API_DECODE_MESSAGE to decode the event message. The event message class and type can be retrieved directly using the Event Message Format described previously. The information contained in the event message record header is in native OpenVMS format.

To process Sterling Connect:Direct for OpenVMS message identifier strings, the calling process must open the Sterling Connect:Direct for OpenVMS message file by calling the NDM_EVENT_API_MSGFILE_OPEN routine. This routine would normally be called only once at process initialization before subsequent event message processing. However, this is not enforced and the Sterling Connect:Direct for OpenVMS message file may be opened/closed at any time and as many times as necessary.

The calling process obtains the text associated with a Sterling Connect:Direct for OpenVMS message identifier string by calling the NDM_EVENT_API_MSGFILE_DISPLAY routine. This routine invokes the caller's action routine for each line of text associated with the message identifier.

The calling process closes the Sterling Connect:Direct for OpenVMS message file by calling the NDM_EVENT_API_MSGFILE_CLOSE routine.

Note: The Sterling Connect:Direct for OpenVMS message file is not a standard OpenVMS message file.

Currently Defined Routines

The following routines are currently defined:

- NDM_EVENT_API_RECEIVE_STREAM
- NDM_EVENT_API_DECODE_MESSAGE
- NDM_EVENT_API_DECODE_ITEM
- NDM_EVENT_API_WRITE_MESSAGE
- NDM_EVENT_API_MSGFILE_OPEN
- NDM_EVENT_API_MSGFILE_DISPLAY
- NDM_EVENT_API_MSGFILE_CLOSE
- NDM_EVENT_API_GET_VERSION

The event message API routines obey OpenVMS calling procedure standards. All routines return system and library conditions value in R0 and serious errors are signalled as well as returned. Refer to Appendix A, *Event Logging Facility-Event Information* for a list of event information.

Receiving the Event Message Stream

The NDM_EVENT_API_RECEIVE_STREAM (Receive the Event Message Stream) routine causes the process to become an event logging monitor.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_RECEIVE_STREAM routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_RECEIVE_STREAM	action_procedure
	action_context
	reserved_1
	reserved_2

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_RECEIVE_STREAM routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for each NDM_EVENT_API_RECEIVE_STREAM routine argument.

action_procedure
the address of a user-supplied action routine called when an event message is received.

OpenVMS Usage:	procedure
type:	procedure value
access:	function call (before return)
mechanism:	by value

action_context
a user-supplied argument that NDM_CSX_API_SCRIPT_TERM_NOTIFY passes to the action procedure. The method used to pass **action_context** to NDM_EVENT_API_RECEIVE_STREAM is also used to pass it to the action routine. If this argument is omitted, a zero is passed by value.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Using the NDM_EVENT_API_RECEIVE_STREAM Routine

NDM_EVENT_API_RECEIVE_STREAM enables the calling process to receive all event messages received by the Sterling Connect:Direct for OpenVMS event logger process. The event action routine is called for every message that is received.

Call Format for the Action Routine

The following table shows the calling format and arguments for the action routine. A description of each argument follows the routine format.

Routine	Arguments
action_procedure	action_context
	message
	sequence
	class
	type
	replay
	reserved_1
	reserved_2

Arguments

The following are the arguments for the action routine:

action_context
value passed by NDM_EVENT_API_RECEIVE_STREAM to the action routine. The same passing method used to pass **action_context** to NDM_EVENT_API_RECEIVE_STREAM is used by NDM_EVENT_API_RECEIVE_STREAM to pass **action_context** to the action routine.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

message
the event message as received from the Sterling Connect:Direct for OpenVMS event logger process. The message argument is the address of a descriptor that points to a buffer containing the event message.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

sequence
the received message counter sequence number of the message.

OpenVMS Usage:	user_arg
type:	longword (signed)
access:	read only
mechanism:	by value

class
specifies the event message class.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	read only
mechanism:	by value

type
specifies the event message type.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	read only
mechanism:	by value

replay
specifies whether the event is a replay as follows:

0-Original Message
1-Replayed Message

OpenVMS Usage:	user_arg
type:	longword (signed)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_2
laceholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Notification was successfully enabled.
SS$_BADPARAM	A reserved argument was not specified or is not zero.
SS$_UNSUPPORTED	Sterling Connect:Direct for OpenVMS event logger process not running.

Note: Any condition value returned by $ASSIGN, $QIO, $CREMBX, LIB$GET_VM, $GETDVI, LIB$GET_EF, LIB$FREE_VM is also stored in register zero (0).

Decoding Event Messages

The NDM_EVENT_API_DECODE_MESSAGE (Decode Event Message) routine decodes an event message into its constituent header and data information.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_DECODE_MESSAGE routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_DECODE_MESSAGE	message
	[time]
	[class]
	[type]
	[node]
	[name]
	[name_length]
	[data_procedure]
	[data_context]
	reserved_1
	reserved_2
	reserved_3

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_DECODE_MESSAGE routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Argument

The following are descriptions for each NDM_EVENT_API_DECODE_MESSAGE routine argument:

message
the address of a descriptor that points to a buffer containing the event message.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

time
the address of a quadword to receive the system time of the event.

OpenVMS Usage:	date_time
type:	quadword (signed)
access:	write only
mechanism:	by reference

class
the address of a word to receive the event message class.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

type
the address of a word to receive the event message type.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

node
the site-specified source node context value of the event message.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	write only
mechanism:	by reference

name
the address of descriptor that points to a character string that receives the descriptive name associated with the event. An event name can be up to 64 characters.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

name_length
the address of a word that receives the length of the descriptive name.

OpenVMS Usage:	ser_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

data_procedure
the address of a user-supplied action routine that NDM_EVENT_API_DECODE_MESSAGE calls for each message data item found.

OpenVMS Usage:	procedure
type:	procedure value
access:	function call (before return)
mechanism:	by value

data_context
user-supplied argument that NDM_EVENT_API_DECODE_MESSAGE passes to the data action procedure. The same method used to pass data_context to NDM_EVENT_API_DECODE_MESSAGE is also used to pass it to the data action routine. If this argument is omitted, a zero is passed by value.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	ongword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_3
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Using the NDM_EVENT_API_DECODE_MESSAGE Routine

NDM_EVENT_API_DECODE_MESSAGE decodes an event message returning the message header fields directly and returning the message data indirectly by means of an action routine. The action routine is called for each data item present in the message. If the low-bit of the status returned by the action routine is clear, no further data items are decoded.

Call Format for Action Routine

The following table shows the calling format and arguments for the action routine. A description of each argument follows the routine format.

Routine	Arguments
data_procedure	data_context
	message
	sequence
	item_code

Routine	Arguments
	item_name
	item_type
	item_data
	item_value
	item_desc
	reserved

Arguments

The following are the arguments for the action routine:

data_context
value passed by NDM_EVENT_API_DECODE_MESSAGE action routine. The same passing method used to pass **data_context** to NDM_EVENT_API_DECODE_MESSAGE is used by NDM_EVENT_API_DECODE_MESSAGE to pass **data_context** to the action routine.

OpenVMS Usage: user_arg
type: longword (unsigned)
access: read only
mechanism: by value

message
the event message as received from the Sterling Connect:Direct for OpenVMS event logger process. The message argument is the address of a descriptor that points to a buffer containing the event message.

OpenVMS Usage: char_string
type: character string
access: read only
mechanism: by descriptor - fixed length string descriptor

sequence
the sequence number of the data item in the event message.

OpenVMS Usage: user_arg
type: longword (signed)
access: read only
mechanism: by value

item_code
specifies the binary identifier code of the data item.

OpenVMS Usage: user_arg
type: longword (unsigned)
access: read only
mechanism: by value

item_name
the address of a descriptor that points to a string containing the name of the data item.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

item_type
specifies the data item type.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

item_data
the address of a descriptor that points to the binary data item.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

item_value
the address of a descriptor that points to a string containing the converted ASCII value of the data item.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

item_desc
the address of a descriptor that points to a string containing the descriptive text associated with the data item.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

reserved
placeholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Notification was successfully enabled.
SS$_BADPARAM	A reserved argument was not specified or is not zero.
SS$_BADCONTEXT	Invalid or corrupted message specified.

Note: Any condition value returned by LIB$ANALYZE_SDESC is also stored in register zero (0).

Decoding Event Data Items

The NDM_EVENT_API_DECODE_ITEM (Decode Event Data Item) routine locates and decodes an event message data item.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_DECODE_ITEM routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_DECODE_ITEM	message
	[sequence]
	item_code
	[name]
	[name_length]
	[item_type]
	[item_data]
	[data_length]
	[item_value]
	[value_length]
	[item_desc]
	[desc_length]
	reserved_1
	reserved_2

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_DECODE_ITEM routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for each NDM_EVENT_API_DECODE_ITEM routine argument:

message
the address of a descriptor that points to a buffer containing the event message.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

sequence
the address of a longword that receives the sequence number of the item in the event message.

OpenVMS Usage:	user_arg
type:	longword (signed)
access:	write only
mechanism:	by reference

item_code
the binary item code identifier value.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

name
the address of descriptor that points to a character string that receives the name of the item. An item name can be up to 31 bytes.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

name_length
the address of a word that receives the length of the item name.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

item_type
the address of a longword that receives the binary item type.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	write only
mechanism:	by reference

item_data
the address of descriptor that points to a character string that receives the binary item data. A data item can be up to 255 bytes.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

data_length
the address of a word that receives the length of the binary data item.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

item_value
the address of descriptor that points to a character string that receives the converted ASCII value of the binary data item. The data item value can be up to 255 bytes.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

value_length
the address of a word that receives the length of the converted ASCII value of the binary data item.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

item_desc
the address of descriptor that points to a character string that receives the descriptive text associated with the data item. The descriptive text can be up to 64 bytes.

OpenVMS Usage:	char_string
type:	character string
access:	write only
mechanism:	by descriptor

desc_length
the address of a word that receives the length of the descriptive text.

OpenVMS Usage:	user_arg
type:	word (unsigned)
access:	write only
mechanism:	by reference

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Decode was successful.
SS$_ITEMNOTFOUND	Data item was not found.
SS$_BADPARAM	A reserved argument was not specified or is not zero.
SS$_RESULTOVF	Return argument string overflow.
SS$_BADCONTEXT	Invalid or corrupted message specified.

Note: Any condition value returned by LIB$ANALYZE_SDESC is also stored in register zero (0).

Writing User-Defined Event Messages

The NDM_EVENT_API_WRITE_MESSAGE (Write User-Defined Event Messages) routine writes a customer-defined message to the event logger process. A user-defined message is obscure and can range from 0–255 characters in length.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_WRITE_MESSAGE routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_WRITE_MESSAGE	[message]
	reserved_1

Routine	Arguments
	reserved_2

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_WRITE_MESSAGE routine:

OpenVMS usage: cond_value
type: longword (signed)
access: write only
mechanism: by value

Arguments

The following are descriptions for each NDM_EVENT_API_WRITE_MESSAGE routine argument:

message
the address of a descriptor that points to a buffer containing the event message.

OpenVMS Usage: char_string
type: character string
access: read only
mechanism: by descriptor

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage: null_arg
type: longword integer (unsigned)
access: read only
mechanism: by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage: null_arg
type: longword integer (unsigned)
access: read only
mechanism: by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Message successfully written.
SS$_BADPARAM	A reserved argument was not specified or is not zero.

Condition Values	Meaning
SS$_UNSUPPORTED	Sterling Connect:Direct for OpenVMS event logger process was not running or an invalid message length was received.

Note: Any condition value returned by $ASSIGN, QIO, LIBANALYZE_SDESC, $TRNLNM is also stored in register zero (0).

Opening a Sterling Connect:Direct for OpenVMS Message File

The NDM_EVENT_API_MSGFILE_OPEN (Open Message File) routine opens a Sterling Connect:Direct for OpenVMS message file. The returned context identifier is used as an input argument to the NDM_EVENT_API_MSGFILE_CLOSE and NDM_EVENT_MSGFILE_DISPLAY routines. The Sterling Connect:Direct for OpenVMS message file is not a standard OpenVMS message file.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_MSGFILE_OPEN routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_MSGFILE_OPEN	msg_file
	msg_fctx
	reserved_1
	reserved_2
	reserved_3
	reserved_4

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_MSGFILE_OPEN routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for each NDM_EVENT_API_MSGFILE_OPEN routine argument:

message

the address of a descriptor that points to a character string containing the filename of a Sterling Connect:Direct for OpenVMS message file.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

msg_fctx
the address of a longword that receives the Sterling Connect:Direct for OpenVMS message file open context identifier.

OpenVMS Usage:	context
type:	longword (unsigned)
access:	write only
mechanism:	by reference

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_3
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_4
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Open successful.
SS$_BADPARAM	A reserved argument was not specified or is not zero.

Note: Any condition value returned by RMS, LIB$ANALYZE_SDESC, LIB$GET_VM, LIB$FREE_VM is also stored in register zero (0).

Displaying Sterling Connect:Direct Message File Text

The NDM_EVENT_API_MSGFILE_DISPLAY (Display Sterling Connect:Direct Message File Text) routine returns the message file text associated with a specified message identifier string.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_MSGFILE_DISPLAY routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_MSGFILE_DISPLAY	msg_fctx
	msg_key
	msg_procedure
	msg_context
	reserved_1
	reserved_2
	reserved_3
	reserved_4

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_MSGFILE_DISPLAY routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are the arguments for the routine:

msg_fctx

the value of the Sterling Connect:Direct for OpenVMS message file identifier returned by the NDM_EVENT_API_MSGFILE_OPEN routine.

OpenVMS Usage:	context
type:	longword (unsigned)
access:	read only
mechanism:	by value

msg_key
the address of a descriptor that points to a character string containing the message identifier to be found. The message identifier can be up to 8 characters in length.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor

msg_procedure
the address of a user-supplied action routine that NDM_EVENT_API_MSGFILE_DISPLAY calls for each line of associated message file text.

OpenVMS Usage:	procedure
type:	procedure value
access:	function call (before return)
mechanism:	by value

msg_context
user-supplied argument that NDM_EVENT_API_MSGFILE_DISPLAY passes to the message file text action procedure. The same method used to pass **msg_context** to NDM_EVENT_API_MSGFILE_DISPLAY is also used to pass it to message file text action routine. If this argument is omitted, a zero is passed by value.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_3
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_4
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Using the NDM_EVENT_API_MSGFILE_DISPLAY Routine

NDM_EVENT_API_MSGFILE_DISPLAY performs a file lookup using the specified message file identifier and returns the message file text indirectly by means of an action routine. The action routine is called for each line of associated message file text. If the low-bit of the status returned by the action routine is clear, no further message file text lines are returned.

Call Format for Action Routine

The following table shows the calling format and arguments for the action routine. A description of each argument follows the routine format.

Routine	Arguments
msg_procedure	msg_context
	msg_text
	reserved

Arguments

The following are the arguments for the action routine:

msg_context
value passed by NDM_EVENT_API_MSGFILE_DISPLAY action routine. The same passing method used to pass **msg_context** to NDM_EVENT_API_MSGFILE_DISPLAY is used by NDM_EVENT_API_MSGFILE_DISPLAY to pass **msg_context** to the action routine.

OpenVMS Usage:	user_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

msg_text
the address of a descriptor that points to a buffer containing a line of the associated message file text.

OpenVMS Usage:	char_string
type:	character string
access:	read only
mechanism:	by descriptor - fixed length string descriptor

reserved
placeholding argument reserved to Sterling Commerce.

OpenVMS Usage:	null_arg
type:	longword (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Success.
SS$_BADPARAM	A reserved argument was not specified or is not zero.
RMS$_RNF	No such message identifier exists.

Note: Any condition value returned by RMS, LIB$ANALYZE_SDESC is also stored in register zero (0).

Closing a Sterling Connect:Direct for OpenVMS Message File

The NDM_EVENT_API_MSGFILE_CLOSE (Close Message File) routine closes a Sterling Connect:Direct for OpenVMS message file.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_MSGFILE_CLOSE routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_MSGFILE_CLOSE	msg_fctx
	reserved_1
	reserved_2
	reserved_3
	reserved_4

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_MSGFILE_CLOSE routine:

OpenVMS usage:	cond_value
type:	longword (signed)
access:	write only
mechanism:	by value

Arguments

The following are descriptions for the NDM_EVENT_API_MSGFILE_CLOSE routine arguments:

msg_fctx
the value of the Sterling Connect:Direct for OpenVMS message file identifier returned by the NDM_EVENT_API_MSGFILE_OPEN routine.

OpenVMS Usage:	context
type:	longword (unsigned)
access:	read only
mechanism:	by value

reserved_1
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

reserved_2
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage:	null_arg
type:	longword integer (unsigned)
access:	read only
mechanism:	by value

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Success.
SS$_BADPARAM	A reserved argument was not specified or is not zero.

Note: Any condition value returned by RMS, LIB$FREE_VM is also stored in register zero (0).

Obtaining the Current API Version Number

The NDM_EVENT_API_GET_VERSION (Return Version Number) routine returns the API version number.

Reviewing the Format and Arguments

The following table shows the format and arguments for the NDM_EVENT_API_GET_VERSION routine. A description of each argument follows the routine format.

Routine	Arguments
NDM_EVENT_API_GET_VERSION	version
	reserved

Return Values

The following are values returned in register zero (0) for the NDM_EVENT_API_GET_VERSION routine:

OpenVMS usage: cond_value
type: longword (signed)
access: write only
mechanism: by value

Arguments

The following are the descriptions for the NDM_EVENT_API_GET_VERSION routine arguments:

version
the address of a longword that receives the API version number.

OpenVMS Usage: user_arg
type: longword (signed)
access: write only
mechanism: by reference

reserved
placeholding argument reserved to Sterling Commerce. This argument must be specified as zero.

OpenVMS Usage: null_arg
type: longword integer (unsigned)
access: read only
mechanism: by value

Using the NDM_EVENT_API_GET_VERSION Routine

NDM_EVENT_API_GET_VERSION returns the version number in the following form:

```
(major_version_number  * 10000) + (minor_version_number * 100 ).
```

For example, V3.6.00 would be returned as 30600.

Returned Condition Values

The following table shows the condition values returned by the routine and their meaning.

Condition Values	Meaning
SS$_NORMAL	Success.
SS$_BADPARAM	A reserved argument was not specified or is not zero.

Appendix A

Event Logging Facility-Event Information

This appendix lists the following Event Logging Facility event information:

- Event list
- Event class
- Event types
- Event item description
- Event item codes
- Sample event message formats

Event List

The current event list is as follows:

Event	Description
9.9	UserInterface.Could Not Establish Link
11.1	Operational.Server Process Starting
11.2	Operational.Server Process Startup Complete
11.3	Operational.User Interface/DECnet Request Link Established
11.4	Operational.User Interface/DECnet Request Link Lost
11.5	Operational.User Interface Access Complete
11.6	Operational.User Interface Access Failure
11.7	Operational.User Interface Proxy Access Complete
11.8	Operational.User Interface Proxy Access Requested
11.9	Operational.User Interface Proxy Access Failure
11.10	Operational.C:D Script File Submit Request Complete
11.11	Operational.C:D Script File Submit Request Failure
11.12	Operational.Netmap Change Request Received

Event	Description
11.13	Operational.C:D Script Status Change Request Received
11.14	Operational.Session Process Create Failure
11.15	Operational.Session Process Timeout Failure
11.16	Operational.User Interface Access Failure
11.17	Operational.Session Process Terminated Abnormally
11.18	Operational.DECnet Session Process Create Request Received
11.19	Operational.C:D Script File Retry Limit Reached
11.20	Operational.C:D Server Process Diagnostic Message
13.1	Requester.Session Process Create Complete
13.2	Requester.Remote Access Control Complete
13.3	Requester.Remote Access Control Failure
13.5	Requester.Remote Proxy Access Requested
13.8	Requester.Session Link Established
13.9	Requester.Session Link Not Established
13.10	Requester.Session Link Lost
13.14	Requester.File Transmit Started
13.15	Requester.File Transmit Complete
13.16	Requester.File Transmit Failure
13.19	Requester.File Receive Started
13.20	Requester.File Receive Complete
13.21	Requester.File Receive Failure
13.24	Requester.RunTask On Local Node Started
13.25	Requester.RunTask On Local Node Complete
13.26	Requester.RunTask On Local Node Failure
13.29	Requester.RunTask On Remote Node Started
13.30	Requester.RunTask On Remote Node Complete
13.31	Requester.RunTask On Remote Node Failure
13.33	Requester.RunJob On Local Node Started
13.34	Requester.RunJob On Local Node Complete
13.35	Requester.RunJob On Local Node Failure
13.38	Requester.RunJob On Remote Node Started
13.39	Requester.RunJob On Remote Node Complete
13.40	Requester.RunJob On Remote Node Failure
13.42	Requester.Submit On Local Node Started

Event	Description
13.43	Requester.Submit On Local Node Complete
13.44	Requester.Submit On Local Node Failure
13.47	Requester.Submit On Remote Node Started
13.48	Requester.Submit On Remote Node Complete
13.49	Requester.Submit On Remote Node Failure
13.51	Requester.Session User Persona Changed
13.52	Requester.Session Complete
13.53	Requester.C:D Session Diagnostic Message
15.1	Responder.Session Process Create Complete
15.2	Responder.Remote Access Control Complete
15.3	Responder.Remote Access Control/Permitted Node Reject Failure
15.5	Responder.Remote Proxy Access Requested
15.7	Responder.Remote Proxy Access Rejected
15.8	Responder.Session Link Established
15.9	Responder.Session Link Not Established
15.10	Responder.Session Link Lost
15.11	Responder.Session Rejected With Remote Node
15.14	Responder.File Transmit Started
15.15	Responder.File Transmit Complete
15.16	Responder.File Transmit Failure
15.19	Responder.File Receive Started
15.20	Responder.File Receive Complete
15.21	Responder.File Receive Failure
15.24	Responder.RunTask On Local Node Started
15.25	Responder.RunTask On Local Node Complete
15.26	Responder.RunTask On Local Node Failure
15.27	Responder.RunTask On Local Node Rejected
15.33	Responder.RunJob On Local Node Started
15.34	Responder.RunJob On Local Node Complete
15.35	Responder.RunJob On Local Node Failure
15.36	Responder.RunJob On Local Node Rejected
15.42	Responder.Submit On Local Node Started
15.43	Responder.Submit On Local Node Complete
15.44	Responder.Submit On Local Node Failure

Event	Description
15.45	Responder.Submit On Local Node Rejected
15.51	Responder.Session User Persona Changed
15.52	Responder.Session Complete
15.53	Responder.C:D Session Diagnostic Message
129.1	EventProcess.Log File Started
129.2	EventProcess.Log File Error Occurred
129.3	EventProcess.Shutdown Request Received
129.4	EventProcess.C:D Server Process Has Terminated
129.8	EventProcess.Path To Sink/Source Node Complete
129.9	EventProcess.Path To Sink/Source Node Failed
129.10	EventProcess.Path To Sink/Source Node Lost
256.1	Customer.Message Generated By Customer

Event Class Definitions

The following are the current event class definitions:

Event Class	Definition
NDMEVLC_USER_INTERFACE	%X00000009
NDMEVLC_OPERATIONAL	%X0000000B
NDMEVLC_SESSION_REQUESTER	%X0000000D
NDMEVLC_SESSION_RESPONDER	%X0000000F
NDMEVLC_EVENT_FACILITY	%X00000081
NDMEVLC_CUSTOMER_DEFINED	%X00000100

The following table defines which programs use event class:

Event Class	Program
User Interface	User Interface (NDMUI) program
Operational	Server process, utilities
SessionRequester	Outbound Session
SessionResponder	Inbound Session
Event Facility	Event Logger Process

Event Class	Program
Customer	Customer-Defined Messages

Event Types

The following are the currently defined event type codes:

Event Types	Codes
NDMEVLT_PROCESSSTARTUP	%X00000001
NDMEVLT_STARTUPCOMPLETE	%X00000002
NDMEVLT_UILINKCOMPLETE	%X00000003
NDMEVLT_UILINKLOST	%X00000004
NDMEVLT_UIACCESSCOMPLETE	%X00000005
NDMEVLT_UIACCESSFAILURE	%X00000006
NDMEVLT_UIPROXYACCESSCOMPLETE	%X00000007
NDMEVLT_UIPROXYACCESSREQUEST	%X00000008
NDMEVLT_UIPROXYACCESSFAILURE	%X00000009
NDMEVLT_SUBMITREQUESTCOMPLETE	%X0000000A
NDMEVLT_SUBMITREQUESTFAILURE	%X0000000B
NDMEVLT_NETMAPCHANGEREQUESTED	%X0000000C
NDMEVLT_STATUSCHANGEREQUESTED	%X0000000D
NDMEVLT_PROCESSCREATEFAILURE	%X0000000E
NDMEVLT_PROCESSTIMEOUTFAILURE	%X0000000F
NDMEVLT_PROCESSTERMABNORMAL	%X00000011
NDMEVLT_DECNETREQUESTRECEIVED	%X00000012
NDMEVLT_RETRYLIMITREACHED	%X00000013
NDMEVLT_CDSERVERPRCDIAGNOSTIC	%X00000014
NDMEVLT_PROCESSCREATECOMPLETE	%X00000001
NDMEVLT_ACCESSCOMPLETE	%X00000002
NDMEVLT_ACCESSFAILURE	%X00000003
NDMEVLT_PROXYACCESSREQUEST	%X00000005
NDMEVLT_PROXYACCESSREJECTED	%X00000007
NDMEVLT_LINKCOMPLETE	%X00000008
NDMEVLT_LINKFAILURE	%X00000009

Event Types	Codes
NDMEVLT_LINKLOST	%X0000000A
NDMEVLT_SESSIONREJECTED	%X0000000B
NDMEVLT_TRANSMITSTARTED	%X0000000E
NDMEVLT_TRANSMITCOMPLETE	%X0000000F
NDMEVLT_TRANSMITFAILURE	%X00000010
NDMEVLT_RECEIVESTARTED	%X00000013
NDMEVLT_RECEIVECOMPLETE	%X00000014
NDMEVLT_RECEIVEFAILURE	%X00000015
NDMEVLT_RUNTASKLOCALSTARTED	%X00000018
NDMEVLT_RUNTASKLOCALCOMPLETE	%X00000019
NDMEVLT_RUNTASKLOCALFAILURE	%X0000001A
NDMEVLT_RUNTASKLOCALREJECTED	%X0000001B
NDMEVLT_RUNTASKREMOTESTARTED	%X0000001D
NDMEVLT_RUNTASKREMOTECOMPLETE	%X0000001E
NDMEVLT_RUNTASKREMOTEFAILURE	%X0000001F
NDMEVLT_RUNJOBLOCALSTARTED	%X00000021
NDMEVLT_RUNJOBLOCALCOMPLETE	%X00000022
NDMEVLT_RUNJOBLOCALFAILURE	%X00000023
NDMEVLT_RUNJOBLOCALREJECTED	%X00000024
NDMEVLT_RUNJOBREMOTESTARTED	%X00000026
NDMEVLT_RUNJOBREMOTECOMPLETE	%X00000027
NDMEVLT_RUNJOBREMOTEFAILURE	%X00000028
NDMEVLT_SUBMITLOCALSTARTED	%X0000002A
NDMEVLT_SUBMITLOCALCOMPLETE	%X0000002B
NDMEVLT_SUBMITLOCALFAILURE	%X0000002C
NDMEVLT_SUBMITLOCALREJECTED	%X0000002D
NDMEVLT_SUBMITREMOTESTARTED	%X0000002F
NDMEVLT_SUBMITREMOTECOMPLETE	%X00000030
NDMEVLT_SUBMITREMOTEFAILURE	%X00000031
NDMEVLT_SESSIONPERSONACHANGE	%X00000033
NDMEVLT_SESSIONCOMPLETE	%X00000034
NDMEVLT_CDSESSIONDIAGNOSTIC	%X00000035
NDMEVLT_LOGFILESTARTED	%X00000001
NDMEVLT_LOGFILEERROR	%X00000002

Event Types	Codes
NDMEVLT_SHUTDOWNRECEIVED	%X00000003
NDMEVLT_SERVERPRCTERMINATED	%X00000004
NDMEVLT_CUSTOMERDATA	%X00000001

Event Item Description

The following are descriptions of the current event data items:

Event Data Items	Description
NDMEVLIF_DATACOMPRESSION	Data Compression
NDMEVLIF_CHECKPOINTRESTART	Checkpoint/Restart
NDMEVLIF_SESSIONRESTART	Session Restarted
NDMEVLIB_REMOTENODERC	Remote Node RC
NDMEVLIB_LOCALNODERC	Local Node RC
NDMEVLIB_CDDIAGNOSTICRC	Sterling Connect:Direct Diagnostic RC
NDMEVLIL_LOCALNODESTATUS	Local Node System Status Code
NDMEVLIL_LOCALNODESTATUSVALUE	Local Node System Status Value
NDMEVLIL_SESSIONSCRIPTNUMBER	Script File Session Number
NDMEVLIL_REMOTENODEFDBK	Remote Node FDBK
NDMEVLIL_LOCALNODEFDBK	Local Node FDBK
NDMEVLIL_PROCESSIDENTIFIER	OpenVMS Process PID
NDMEVLIL_CDDIAGNOSTICFDBK	Sterling Connect:Direct Diagnostic FDBK
NDMEVLIQ_RECEIVERECORDCOUNT	Local Node Records Received
NDMEVLIQ_RECEIVEBYTECOUNT	Local Node Bytes Received
NDMEVLIQ_TRANSMITRECORDCOUNT	Local Node Records Sent
NDMEVLIQ_TRANSMITBYTECOUNT	Local Node Bytes Sent
NDMEVLIT_TRANSFERSTARTTIME	Transfer Start Time
NDMEVLIT_TRANSFERSTOPTIME	Transfer Stop Time
NDMEVLIS_REQUESTIDENTIFIER	Script File Submit Request Identifier
NDMEVLIS_SESSIONSCRIPTNAME	Script File Session Name
NDMEVLIS_SESSIONSCRIPTSTEP	Script File Session Stepname
NDMEVLIS_SESSIONSCRIPTFILE	Script File Session Pathname
NDMEVLIS_SCRIPTSUBMITUSERNAME	Script File Submit Username

Event Data Items	Description
NDMEVLIS_SESSIONLINKTYPE	Script File Session Protocol Type
NDMEVLIS_SESSIONDECNETNODE	Remote DECnet Node Name
NDMEVLIS_SESSIONDECNETTASK	Remote DECnet Node Task
NDMEVLIS_SESSIONTCPADDRESS	Remote TCP/IP Node Address
NDMEVLIS_SESSIONTCPPORTNUMBER	Remote TCP/IP Node Port Number
NDMEVLIS_SESSIONSNALU0SESSID	SNALU0 Session Number
NDMEVLIS_SESSIONSNALU0APPLID	SNALU0 Session APPLID
NDMEVLIS_REMOTENODENAME	Remote C:D Node Name
NDMEVLIS_REMOTENODEUSER	Remote Node User Name
NDMEVLIS_REMOTENODEMSGID	Remote Node MSGID
NDMEVLIS_LOCALNODENAME	Local C:D Node Name
NDMEVLIS_LOCALNODEUSER	Local Node User Name
NDMEVLIS_LOCALNODEMSGID	Local Node MSGID
NDMEVLIS_SOURCEFILENAME	Source File Name
NDMEVLIS_DESTINATIONFILENAME	Destination File Name
NDMEVLIS_PROCESSNAME	OpenVMS Process Name
NDMEVLIS_RELEASEVERSIONSTRING	Product Release Version String
NDMEVLIS_DECNETDEVICENAME	DECnet Object/Link Device Name
NDMEVLIS_PROCESSNODENAME	OpenVMS Process Node Name
NDMEVLIS_REQUESTERUSERNAME	Username Of Requester
NDMEVLIS_REQUESTERUSERNODE	Nodename Of Requester
NDMEVLIS_REQUESTERACTION	Action Requested By User
NDMEVLIS_SESSIONSNALU0GATEWAY	SNA LU0 Gateway/Access Name
NDMEVLIS_CDDIAGNOSTICMSGID	Sterling Connect:Direct Diagnostic MSGID
NDMEVLIV_CUSTOMERDEFINEDDATA	Customer Data

Event Item Codes

The following is the current list of event data item codes:

Event Data Item	Code
NDMEVLIM_	%XF0000000
NDMEVLIF_	%X10000000

Event Data Item	Code
NDMEVLIB_	%X20000000
NDMEVLIW_	%X30000000
NDMEVLIL_	%X40000000
NDMEVLIQ_	%X50000000
NDMEVLIT_	%X60000000
NDMEVLIS_	%X70000000
NDMEVLIV_	%X80000000
NDMEVLIL_LOCALNODESTATUS	%X40000002
NDMEVLIL_LOCALNODESTATUSVALUE	%X40000004
NDMEVLIS_REQUESTIDENTIFIER	%X70000006
NDMEVLIS_SESSIONSCRIPTNAME	%X70000008
NDMEVLIL_SESSIONSCRIPTNUMBER	%X4000000A
NDMEVLIS_SESSIONSCRIPTSTEP	%X7000000C
NDMEVLIS_SESSIONSCRIPTFILE	%X7000000E
NDMEVLIS_SCRIPTSUBMITUSERNAME	%X70000010
NDMEVLIS_SESSIONLINKTYPE	%X70000012
NDMEVLIS_SESSIONDECNETNODE	%X70000014
NDMEVLIS_SESSIONDECNETTASK	%X70000016
NDMEVLIS_SESSIONTCPADDRESS	%X70000018
NDMEVLIS_SESSIONTCPPORTNUMBER	%X7000001A
NDMEVLIS_SESSIONSNALU0SESSID	%X7000001C
NDMEVLIS_SESSIONSNALU0APPLID	%X7000001E
NDMEVLIS_REMOTENODENAME	%X70000020
NDMEVLIS_REMOTENODEUSER	%X70000022
NDMEVLIB_REMOTENODERC	%X20000024
NDMEVLIS_REMOTENODEMSGID	%X70000026
NDMEVLIL_REMOTENODEFDBK	%X40000028
NDMEVLIS_LOCALNODENAME	%X7000002A
NDMEVLIS_LOCALNODEUSER	%X7000002C
NDMEVLIB_LOCALNODERC	%X2000002E
NDMEVLIS_LOCALNODEMSGID	%X70000030
NDMEVLIL_LOCALNODEFDBK	%X40000032
NDMEVLIS_SOURCEFILENAME	%X70000034
NDMEVLIS_DESTINATIONFILENAME	%X70000036

Event Data Item	Code
NDMEVLIQ_RECEIVERECORDCOUNT	%X50000038
NDMEVLIQ_RECEIVEBYTECOUNT	%X5000003A
NDMEVLIQ_TRANSMITRECORDCOUNT	%X5000003C
NDMEVLIQ_TRANSMITBYTECOUNT	%X5000003E
NDMEVLIF_DATACOMPRESSION	%X10000040
NDMEVLIF_CHECKPOINTRESTART	%X10000042
NDMEVLIS_PROCESSNAME	%X70000044
NDMEVLIL_PROCESSIDENTIFIER	%X40000046
NDMEVLIS_RELEASEVERSIONSTRING	%X70000048
NDMEVLIS_DECNETDEVICENAME	%X7000004A
NDMEVLIS_PROCESSNODENAME	%X7000004C
NDMEVLIS_REQUESTERUSERNAME	%X7000004E
NDMEVLIS_REQUESTERUSERNODE	%X70000050
NDMEVLIS_REQUESTERACTION	%X70000052
NDMEVLIS_SESSIONSNALU0GATEWAY	%X70000054
NDMEVLIB_CDDIAGNOSTICRC	%X20000056
NDMEVLIS_CDDIAGNOSTICMSGID	%X70000058
NDMEVLIL_CDDIAGNOSTICFDBK	%X4000005A
NDMEVLIF_SESSIONRESTART	%X1000005C
NDMEVLIT_TRANSFERSTARTTIME	%X6000005E
NDMEVLIT_TRANSFERSTOPTIME	%X60000060
NDMEVLIV_CUSTOMERDEFINEDDATA	%X80001000

Sample Event Message Formats

The following are examples of selected event messages that specify the typical data items associated with the message.

Event 11.10 Operational.C:D Script File Submit Request Complete

NDMEVLIS_REQUESTERUSERNAME
NDMEVLIS_SESSIONSCRIPTFILE
NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME
NDMEVLIL_SESSIONSCRIPTNUMBER

Event 13.15 Requester.File Transmit Complete

NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME

NDMEVLIL_SESSIONSCRIPTNUMBER
NDMEVLIT_TRANSFERSTARTTIME
NDMEVLIT_TRANSFERSTOPTIME
NDMEVLIS_SOURCEFILENAME
NDMEVLIS_DESTINATIONFILENAME
NDMEVLIS_REMOTENODENAME
NDMEVLIS_REQUESTERUSERNAME
NDMEVLIQ_TRANSMITBYTECOUNT
NDMEVLIQ_TRANSMITRECORDCOUNT

Event 13.16 Requester.File Transmit Failure

NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME
NDMEVLIL_SESSIONSCRIPTNUMBER
NDMEVLIT_TRANSFERSTARTTIME
NDMEVLIT_TRANSFERSTOPTIME
NDMEVLIS_SOURCEFILENAME
NDMEVLIS_DESTINATIONFILENAME
NDMEVLIS_REMOTENODENAME
NDMEVLIS_REQUESTERUSERNAME
NDMEVLIQ_TRANSMITBYTECOUNT
NDMEVLIQ_TRANSMITRECORDCOUNT
NDMEVLIB_REMOTENODERC
NDMEVLIS_REMOTENODEMSGID
NDMEVLIL_REMOTENODEFDBK

Event 15.20 Responder.File Receive Complete

NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME
NDMEVLIL_SESSIONSCRIPTNUMBER
NDMEVLIT_TRANSFERSTARTTIME
NDMEVLIT_TRANSFERSTOPTIME
NDMEVLIS_SOURCEFILENAME
NDMEVLIS_DESTINATIONFILENAME
NDMEVLIS_REMOTENODENAME
NDMEVLIS_REQUESTERUSERNAME
NDMEVLIQ_RECEIVEBYTECOUNT
NDMEVLIQ_RECEIVERECORDCOUNT

Event 15.20 Responder.File Receive Failure

NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME
NDMEVLIL_SESSIONSCRIPTNUMBER
NDMEVLIT_TRANSFERSTARTTIME
NDMEVLIT_TRANSFERSTOPTIME
NDMEVLIS_SOURCEFILENAME
NDMEVLIS_DESTINATIONFILENAME
NDMEVLIS_REMOTENODENAME
NDMEVLIS_REQUESTERUSERNAME
NDMEVLIQ_RECEIVEBYTECOUNT
NDMEVLIQ_RECEIVERECORDCOUNT
NDMEVLIB_LOCALNODERC
NDMEVLIS_LOCALNODEMSGID
NDMEVLIL_LOCALNODEFDBK

Event 15.53 Responder.C:D Session Diagnostic Message

NDMEVLIS_REQUESTIDENTIFIER
NDMEVLIS_SESSIONSCRIPTNAME
NDMEVLIL_SESSIONSCRIPTNUMBER
NDMEVLIS_REMOTENODENAME
NDMEVLIB_CDDIAGNOSTICRC
NDMEVLIL_CDDIAGNOSTICFDBK
NDMEVLIS_CDDIAGNOSTICMSGID

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual

Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

1623-14, Shimotsuruma, Yamato-shi

Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA__95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available. This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are ficticious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© IBM 2011. Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. 2011.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Glossary

A

Adjacent Node

An adjacent node is an entry in the Network Map that defines a Sterling Connect:Direct node with which the local Sterling Connect:Direct node can communicate. The adjacent node is also referred to as a remote node.

Application Program Interface (API)

The Application Program Interface (API) is a Sterling Connect:Direct component that accepts commands and places them in an executable format.

C

Checkpoint Restart

The Checkpoint Restart feature eliminates the need to re-transmit an entire file in the event of a transmission failure. If a copy procedure is interrupted, Sterling Connect:Direct restarts that copy at the last checkpoint.

Command Line Interface

The Command Line Interface is a Sterling Connect:Direct interface that allows users to submit Sterling Connect:Direct Processes and commands from their native command line environment.

Commands

Sterling Connect:Direct commands initiate and monitor activity within the Sterling Connect:Direct system.

I

Interactive Mode

The interactive mode of operation allows you to input multiple commands with one invocation of the user interface (NDMUI).

L

Local Node

The local node is the Sterling Connect:Direct server.

N

NDMUI

is the command that invokes the user interface. The user interface is one of the windows through which you communicate with Sterling Connect:Direct for OpenVMS.

Network Map (Netmap)

The Network Map (netmap) is a file that identifies all valid Sterling Connect:Direct nodes in the network. One Network Map is associated with each Sterling Connect:Direct local node. The netmap has one entry for each of the other Sterling Connect:Direct nodes to which the local Sterling Connect:Direct node communicates. The netmap entries also contain the rules or protocol that the nodes adhere to when communicating.

Node

A node is any site in a network from which information distribution can be initiated.

Noninteractive Mode

The noninteractive mode of operation allows you to issue a single command in one invocation of the NDMUI.

P

Passive Connect Session

The passive connect session is a session created for a request from a remote node and is for the SNA communications connection only.

Primary Node (PNODE)

The Primary Node (PNODE) is the Sterling Connect:Direct node on which the Process is submitted. The primary node can also be referred to as the controlling node or initiating node, but should not necessarily be interpreted as the sending node, since PNODE can be the receiver. In every Process, there is one PNODE and one SNODE specified. The submitter of a Process is always the PNODE.

Process

A Process is a series of statements that initiate Sterling Connect:Direct activity, such as copying files, running jobs, and so on.

Process Statements

Process Statements are instructions for transferring files, running operating system jobs, executing programs, or submitting other Sterling Connect:Direct Processes. They are used to build a Sterling Connect:Direct Process.

R

Remote Node

A remote node is an entry in the Network Map that defines a Sterling Connect:Direct node with which the local Sterling Connect:Direct node can communicate. The remote node is also referred to as an adjacent node.

Retry Interval

The retry interval is the interval at which retries are performed if the Process encounters nonfatal errors.

S

Secondary Node (SNODE)

The secondary node (SNODE) is the Sterling Connect:Direct node that interacts with the primary node (PNODE) during Process execution. SNODE can also be referred to as the participating (non-controlling) or partner node. Every Process has one PNODE and one SNODE.

Server

The server accepts command requests, communicates with the session manager when work is placed in the Transmission Control Queue (TCQ), and accepts session establishment requests from remote nodes.

Session Manager

The session manager establishes communication sessions, performs standard session management functions, and executes Processes.

Statistics File

The statistics file holds Sterling Connect:Direct statistics records that document the history of a Process.

Statistics Facility

The Sterling Connect:Direct Statistics Facility records Sterling Connect:Direct activities.

T

Transmission Control Queue (TCQ)

The Transmission Control Queue (TCQ) holds information about Sterling Connect:Direct Processes that are currently executing or scheduled to execute in the future.

U

User Interface (NDMUI or UI)

The communications process used to interface with the server. Commands are issued through the user interface when you want to work with the Sterling Connect:Direct product.

Index

Symbols

A

C

D

E

Q

R

S

T

U

V

W