

# HP Operations Orchestration Software

Software Version: 9.00.05

## *TN3270 Integration Guide*

Document Release Date: April 2011

Software Release Date: April 2011



## Legal Notices

### Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

### Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

### Copyright Notices

© Copyright 2010-2011 Hewlett-Packard Development Company, L.P.

### Trademark Notices

For information on open-source and third-party software acknowledgements, see in the documentation set for this release, Open-Source and Third-Party Software Acknowledgements (3rdPartyOpenNotices.pdf).

# On the Web: Finding OO support and documentation

There are two Web sites where you can find support and documentation, including updates to OO Help systems, guides, and tutorials:

- The OO Support site
- HP Live Network

## Support

Documentation enhancements are a continual project at Hewlett-Packard Software. You can obtain or update the HP OO documentation set and tutorials at any time from the HP Software Product Manuals Web site. You will need an HP Passport to log in to the Web site.

### To obtain HP OO documentation and tutorials

1. Go to the HP Software Product Manuals Web site (<http://support.openview.hp.com/selfsolve/manuals>).
2. Log in with your HP Passport user name and password.

OR

If you do not have an HP Passport, click **New users – please register** to create an HP Passport, then return to this page and log in.

If you need help getting an HP Passport, see your HP OO contact.

3. In the **Product** list box, scroll down to and select **Operations Orchestration**.
4. In the **Product Version** list, click the version of the manuals that you're interested in.
5. In the **Operating System** list, click the relevant operating system.
6. Click the **Search** button.
7. In the **Results** list, click the link for the file that you want.

## HP Live Network

For support information, including patches, troubleshooting aids, support contract management, product manuals and more, visit the following site: <https://www.www2.hp.com/>.

This is the **HP Live Network** Web page. To sign in:

1. Click **Login**.
2. On the **HP Passport sign-in** page, enter your HP Passport user ID and password and then click **Sign-in**.
3. If you do not already have an HP Passport account, do the following:
  - a. On the **HP Passport sign-in** page, click **New user registration**.
  - b. On the **HP Passport new user registration** page, enter the required information and then click **Continue**.
  - c. On the confirmation page that opens, check your information and then click **Register**.
  - d. On the **Terms of Service** page, read the Terms of use and legal restrictions, select the **Agree** button, and then click **Submit**.
4. On the **HP Live Network** page, click **Operations Orchestration Community**.

**The Operations Orchestration Community** page contains links to announcements, discussions, downloads, documentation, help, and support.

**Note:** Contact your OO contact if you have any difficulties with this process.

## In OO: How to find Help, PDFs, and tutorials

The HP Operations Orchestration software (HP OO) documentation set is made up of the following:

- Help for Central

Central Help provides information to the following:

- Finding and running flows
- For HP OO administrators, configuring the functioning of HP OO
- Generating and viewing the information available from the outcomes of flow runs

The Central Help system is also available as a PDF document in the HP OO home directory, in the \Central\docs subdirectory.

- Help for Studio

Studio Help instructs flow authors at varying levels of programming ability.

The Studio Help system is also available as a PDF document in the HP OO home directory, in the \Studio\docs subdirectory.

- Animated tutorials for Central and Studio

HP OO tutorials can each be completed in less than half an hour and provide basic instruction on the following:

- In Central, finding, running, and viewing information from flows
- In Studio, modifying flows

The tutorials are available in the Central and Studio subdirectories of the HP OO home directory.

- Self-documentation for operations and flows in the Accelerator Packs and ITIL folders

Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

# Table of Contents

Warranty ..... ii

Restricted Rights Legend ..... ii

Trademark Notices ..... ii

On the Web: Finding OO support and documentation ..... iii

    Support ..... iii

    HP Live Network ..... iii

In OO: How to find Help, PDFs, and tutorials ..... iv

Overview of the TN3270 integration ..... 1

    Use cases and scenarios ..... 1

Installation and configuration instructions ..... 1

Versions ..... 2

Architecture ..... 2

TN3270 operation and flow infrastructure ..... 2

Common inputs in the integration ..... 3

Common results in the integration ..... 3

Operation and flow specifics ..... 5

    Samples ..... 5

        Menu Capture Sample ..... 5

Capture All Fields .....	6
Capture Field .....	6
Capture Screen .....	7
Connect.....	7
Character sets.....	9
Disconnect .....	10
Execute Command.....	11
Move Cursor .....	11
Move To Field .....	11
Read Buffer .....	12
Special characters.....	12
Send Characters.....	13
Special characters.....	14
Send PA Key .....	14
Send PF Key .....	15
Send SysReq.....	15
Snap .....	15
Snap commands .....	15
Wait .....	16
Wait commands.....	16
<b>Troubleshooting.....</b>	<b>17</b>
General troubleshooting procedures and tools .....	18
Connect to the host with a terminal emulator.....	18
Usage of the Connect operation .....	18
<b>Security .....</b>	<b>18</b>
RAS to SSH relay host.....	18
SSH relay host to TN3270 host .....	18
<b>Tools.....</b>	<b>18</b>

# Overview of the TN3270 integration

The TN3270 integration contains operations that are based on the scripting component of the open source 3270 terminal emulator x3270. The operations provide convenient wrappers to the major functionality of the s3270 program, with one operation, **Execute Command**, which you can use to implement functions not wrapped by the other operations.

With the TN3270 operations you can create flows to interact with and gather data from TN3270 enabled mainframe hosts.

## Use cases and scenarios

The following are the major use cases for the set of TN3270 operations, with specific operations used to implement them listed.

1. Data gathering:
  - Capture All Fields
  - Capture Field
  - Capture Screen
  - Read Buffer
  - Snap
2. System control:
  - Move Cursor
  - Move To Field
  - Send Characters
  - Send PF Key
  - Send PA Key
  - Send SysReq
  - Snap
  - Wait
  - Execute Command

## Installation and configuration instructions

In order to use the TN3270 operations, you must have a host with SSH access capable of running the s3270 open source application. S3270 is a component of the X windows 3270 terminal emulator x3270. For a complete list of systems on which s3270 can run, see the x3270 Web site at <http://x3270.bgp.nu/>.

Since most Linux distributions have packaged versions of x3270 which include s3270, a Linux host is probably the best choice for this intermediary server.

## Versions

Operations Orchestration Version	S3270 Version
9.00.05	3.3+

## Architecture

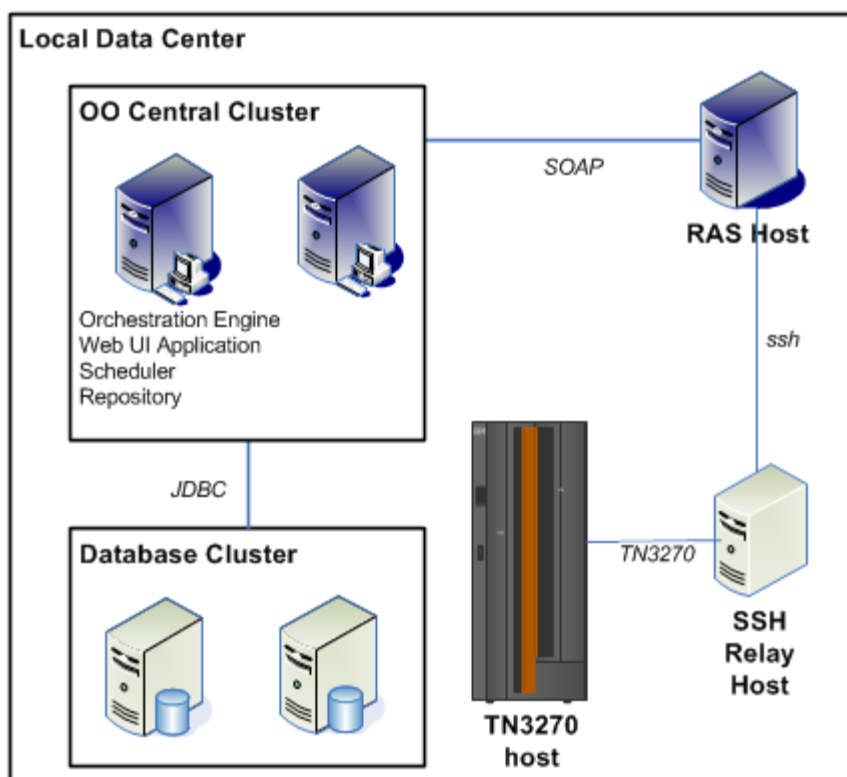


Figure 1 – TN3270 architecture

## TN3270 operation and flow infrastructure

The TN3270 operations include the following operations in the OO Studio Library/Operations/TN3270/ folder.



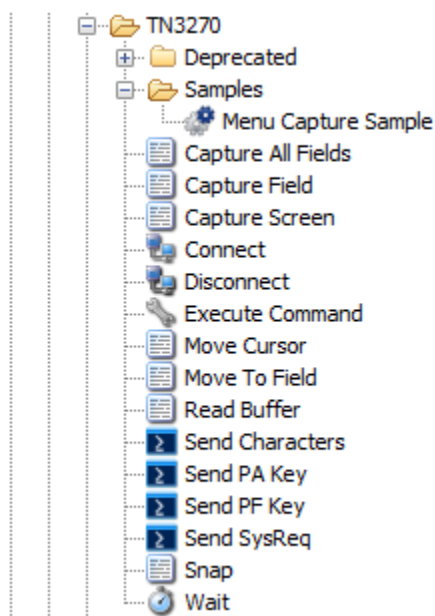


Figure 2 – TN3270 operation and flow infrastructure

## Common inputs in the integration

OO operations and flows use inputs to specify how they obtain the data that they need and when the data is obtained. The following inputs are used by every operation and flow in the TN3270 integration except the **Connect** operation.

### session

The session identifier returned by the **Connect** operation, used to identify the TN3270 connection to use with the operation.

## Common results in the integration

OO operations and flows use results to return data to the caller. Every command executed by the s3270 program returns a status line which is parsed into the following results that returned by every operation and flow in the TN3270 integration.

**Note:** These results are omitted from the operation and flow descriptions in the *Operation and flow specifics* section below since they are the same for every operation and flow. However, if one of the results is important to a particular operation or flow, it is listed in the operation or flow's *Results* section.

### fieldNumber

The comma-separated list of identifiers for each field returned.

### fieldRow

The comma-separated list that contains the start row for each field.

### fieldColumn

The comma-separated list that contains the start column for each field.

**fieldValue**

The comma-separated list that contains the ASCII string contained in each field.

**keyboardState**

Specifies the state of the keyboard. The values are:

- The letter **U** if the keyboard is unlocked.
- The letter **L** if the keyboard is locked while waiting for a response from the host or if you are not connected to a host.
- The letter **E** if the keyboard is locked because of an operator error (such as a field overflow or protected field).

**screenFormatting**

Specifies the formatting state of the screen. The values are:

- The letter **F** if the screen is formatted.
- The letter **U** if the screen is unformatted or in NVT mode.

**fieldProtection**

Specifies the state of the cursor contained in a field. The values are:

- The letter **P** if the field containing the cursor is protected.
- The letter **U** if the field containing the cursor is unprotected or unformatted.

**connectionState**

Specifies the connection state. The values are:

- The string **C(hostname)** if you are connected to a host.
- The letter **N** if you are not connected.

**emulatorMode**

Specifies the emulator mode. The values are:

- The letter **I** if the emulator is connected in 3270 mode.
- The letter **L** if the emulator connected in NVT line mode.
- The letter **C** if the emulator is connected in NVT character mode.
- The letter **P** if the emulator is connected in unnegotiated mode (no BIND active from the host).
- The letter **N** if the emulator is not connected.

**modelNumber**

The model number. The values are 2-5. See the **Connect** operation for details on the model number.

**numRows**

The current number of rows defined on the screen. The host can request that the emulator use a 24x80 screen, so this number may be smaller than the maximum number of rows possible with the current model.

**numCols**

The current number of columns defined on the screen. The host can request that the emulator use a 24x80 screen, so this number may be smaller than the maximum number of columns possible with the current model.

**cursorRow**

The current cursor row (zero-origin).

**cursorColumn**

The current cursor column (zero-origin).

## **windowId**

The X window identifier for the main x3270 window. The format of the identifier is a hexadecimal string preceded by **0x**. For s3270, ws3270 and c3270, this result is a zero.

## **executionTime**

The time that it took for the host to respond to the previous command, in the format *seconds.milliseconds*. If the previous command did not require a host response, this result is a dash.

# Operation and flow specifics

This section describes the details specific to each operation and flow in the TN3270 integration, including their purpose, suggested usage, special inputs and their meanings, and any outputs that are in addition to those shown in the *Common results in the integration* section.

## Samples

### Menu Capture Sample

The **Menu Capture Sample** flow demonstrates how to use the TN3270 operations to log in to a host system, navigate a menu tree, capture some data, and log back out of the system.

It also demonstrates the proper use of the **Connect** and **Disconnect** operations. Special note should be made of where the **Disconnect** operation appears in this flow. There is no execution path after the **Connect** operation has been run, that can exit this flow without going through the **Disconnect** operation. It is very important that this pattern be followed with all flows that use the TN3270 operations. Failure to disconnect an open session could leave resources open on your TN3270 host and the SSH relay host, and potentially cause failures in subsequent steps.

Also, pay close attention to the **Connect** step of this flow. When you add the **Connect** operation to a flow, you must add a result to the **Outputs** tab of the **Connect** operation. This result must be named **session** and be assigned from the step result **session**. If this result is not added to the flow step, the subsequent operations will fail with a message that states the session could not be retrieved from the context.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

## **host**

The TN3270 host to which to open a session.

## **hostUsername**

The TN3270 host username.

## **hostPassword**

The TN3270 host password.

## **sshRelayHost**

The host name of the SSH server that runs the s3270 program.

## **sshRelayUsername**

The username for the SSH host.

## **sshRelayPassword**

The password for the SSH user.

## menuSelection

The keys used to select the desired menu.

All of the operation's results except the following are described in [Common results in the integration](#).

## menuContents

The ASCII screen dump from the requested menu.

# Capture All Fields

The **Capture All Fields** operation captures each field on the screen. The fields are returned in four comma-separated lists:

- Field identifier
- Starting row
- Starting column
- Contents represented as an ASCII string

**Note:** This operation requires that the screen be formatted. If the screen is not formatted, the operation fails.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

## session

The s3270 session identifier, obtained from the **Connect** operation.

All of the operation's results except the following are described in [Common results in the integration](#).

## data

The results from the **ReadBuffer** command, used to gather the field information.

# Capture Field

The **Capture Field** operation captures the contents of the current field (the field in which the cursor is in currently), in either ASCII or EBCDIC format.

**Note:** This operation requires that the screen be formatted. If the screen is not formatted, the operation fails.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

## captureType

The format of the output data. The format can be either ASCII or EBCDIC. The default is ASCII.

All of the operation's results except the following are described in [Common results in the integration](#).

## data

The contents of current field, in ASCII or EBCDIC format.

## Capture Screen

The **Capture Screen** operation captures the contents of the entire screen, in either ASCII or EBCDIC format.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### captureType

The format of the output data. The valid values are **ASCII** and **EBCDIC**. The default is **ASCII**.

All of the operation's results except the following are described in [Common results in the integration](#).

### data

The contents of entire screen in ASCII or EBCDIC format.

## Connect

The **Connect** operation opens a connection to a TN3270 host for use in subsequent operations.

All other operations in the **TN3270** folder use the session returned by this operation to communicate with the TN3270 host. Therefore, you must add a result named **session** to this operation.

You must close the session that this operation opens before exiting a flow that uses this operation (whether or not the flow succeeds). If you do not close the session, the session stays open. See the Library/Operations/TN3270/Samples/ folder for flows that demonstrate this.

This operation makes an SSH connection to the relay host, and executes the s3270 command on that system, connecting to the TN3270 host.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### host

The host with which to make a TN3270 connection. The full syntax for host is:

`[prefix:]...[LUsername@]hostname[:port]`

You can configure the host connection using one of the following options:

- Add a prefix of **P:** to the hostname (**P:hostname**) to cause the connection to go through the **TELNET-PASSTHRU** service rather than directly to the host.
- Add a prefix of **S:** to the hostname (**S:hostname**) to remove the **extended data stream** option reported to the host.
- Add a prefix of **N:** to the hostname (**N:hostname**) to turn off TN3270E support for the session.
- Add a prefix of **L:** to the hostname (**L:hostname**) to cause x3270 to first create an SSL tunnel to the host and then create a TN3270 session inside the tunnel. (This function is supported only if x3270 was built with SSL/TLS support.) Note that TLS-encrypted sessions using the **TELNET START-TLS** option are negotiated with the host automatically; for these sessions do not use the **L:** prefix.
- Add a specific Logical Unit (LU) name to use to the hostname with an ampersand (**@**) (**LUsername@hostname**). You can add multiple LU names to try, separated by commas (**LUsername1,LUsername2@hostname**). You can also place an empty LU in the list by using an extra comma (**LUsername1,,LUsername2@hostname**).

**Note:** The LU name is used for different purposes on different types of hosts. For example, CICS uses the LU name as the Terminal ID.

You can place the hostname inside square bracket characters ([ ]) to prevent any colon (:) characters in the hostname from being interpreted as part of option prefixes or port numbers. This allows you to use numeric IPv6 addresses as hostnames.

On systems that support the **forkpty** library call, you can replace the hostname with **-e** and a command string. This causes x3270 to connect to a local child process, such as a shell.

The port to connect to defaults to **TELNET**. You can override this with the **-port** option, or by appending a port to the hostname with a colon (:). For compatibility with previous versions of x3270 and with tn3270(1), you can also specify the port as a second separate argument.

## model

The model of 3270 display to be emulated. The default model is **3278-4**. The format of the model name is two parts, either of which you can omit:

- The first part is the base model, either **3278** or **3279**. Base model **3278** specifies a monochrome (green on black) 3270 display; base model **3279** specifies a color 3270 display.
- The second part is the model number, which specifies the number of rows and columns. The default is model number **4**.

Model Number	Columns	Rows
2	80	24
3	80	32
4	80	43
5	132	27

**Note:** Technically, there are no 3279-4 or 3279-5 displays, but most hosts seem to work with them anyway.

## ibmCharacterSet

The EBCDIC host character set used by s3270. See the [Character sets](#) section for details.

## s3270Path

The fully qualified path to the s3270 executable on the remote SSH host.

## Timeout

The timeout in milliseconds for every command in the **TN3270** folder.

## extraParameters

A freeform text field, appended to the end of the s3270 command. You can use this to specify less common options of the s3270 program that do not have their own inputs.

## sshRelayHost

The host name of the SSH server that runs the s3270 program.

## sshRelayUsername

The username for the SSH host.

## sshRelayPassword

The password for the SSH user.

## sshRelayCharacterSet

The character set used to communicate with the SSH host (this is not the character set used between s3270 and the tn3270 host).

## sshRelayNewlineChars

Newline characters to use for communicating with the SSH host.

## sshRelayPkFile

The private key file to use for authentication with the SSH host.

All of the operation's results except the following are described in [Common results in the integration](#).

## session

The session identifier for the TN3270 host connection, used by all other TN3270 operations to identify this connection.

## Character sets

The **-charset** option and the **s3270.charset** resource control the EBCDIC host character set used by s3270. The available sets include those shown in the following table.

Charset Name	Host Code Page	Character Set
belgian	500	iso8859-1
belgian-euro	1148	iso8859-15
bracket	37	iso8859-1
brazilian	275	iso8859-1
chinese-gb18030	1388	iso8859-1 + iso10646-1
cp1047	1047	iso8859-1
cp870	870	iso8859-2
finnish	278	iso8859-1
finnish-euro	1143	iso8859-15
french	297	iso8859-1
french-euro	1147	iso8859-15
german	273	iso8859-1
german-euro	1141	iso8859-15
greek	423	iso8859-7
hebrew	424	iso8859-8
icelandic	871	iso8859-1
icelandic-euro	1149	iso8859-15
italian	280	iso8859-1
italian-euro	1144	iso8859-15
japanese-kana	930	jisx0201.1976-0 + jisx0208.1983-0

Charset Name	Host Code Page	Character Set
japanese-latin	939	jisx0201.1976-0 + jisx0208.1983-0
norwegian	277	iso8859-1
norwegian-euro	1142	iso8859-15
russian	880	koi8-r
simplified-chinese	935	iso8859-1 + gb2312.1980-0
slovenian	870	iso8859-2
spanish	284	iso8859-1
spanish-euro	1145	iso8859-15
thai	1160	iso8859-11 tis620.2529-0
traditional-chinese	937	iso8859-1 + Big5-0
turkish	1026	iso8859-9
uk	285	iso8859-1
uk-euro	1146	iso8859-15
us-euro	1140	iso8859-15
us-intl	37	iso8859-1

The default character set is **bracket**, which is useful for common U.S. IBM hosts that use EBCDIC codes AD and BD for the [ and ] characters, respectively.

**Note:** You can specify any of the host code pages listed above by adding **cp** to the host code page, (for example, **cp037** for host code page 037). Also note that the code pages available for a given version of s3270 are displayed using the **-v** command-line option.

## Disconnect

The **Disconnect** operation disconnects the TN3270 session from the host, and closes the SSH connection to the SSH relay host.

This operation tears down the connection that is made with the **Connect** operation, and must be in every execution path once a session has been opened. Failing to properly disconnect a session, once it has been connected, can cause connection issues with your host system and your SSH relay host.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### session

The S3270 session identifier, obtained from the **Connect** operation.

All of the operation's results are described in [Common results in the integration](#).



## Execute Command

The **Execute Command** operation executes an s3270 command and returns any resulting data.

This operation allows you to execute any command supported by s3270, whether or not it has an operation that wraps it.

For a complete list of commands supported by s3270, and more information on what s3270 can do, read the s3270 documentation at <http://x3270.bgp.nu/documentation.html> or on the Unix man pages for s3270(1) and x3270-script(1) on your SSH relay host. The Unix man program is used to view the online documentation installed with a package. This documentation is commonly referred to as the man pages.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### **command**

The complete command to execute, with all parameters supplied.

All of the operation's results except the following are described in [Common results in the integration](#).

### **data**

Any data returned from the command with the leading **data:** prefix removed. Multi-line data retains its newline characters.

## Move Cursor

The **Move Cursor** operation moves the cursor to the specified row and column on the s3270 screen. Typical uses of this operation include positioning the cursor on the screen and sending characters to that location (with the **Send Characters** operation), or reading the contents of the field under the cursor (with the **Read Field** operation).

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### **row**

The row to which to move the cursor.

### **column**

The column to which to move the cursor.

All of the operation's results are described in [Common results in the integration](#).

## Move To Field

The **Move To Field** operation moves the cursor to the row and column defined by the start of the specified field. You can obtain a list of the fields on the screen with the **Capture All Fields** operation, which returns a list that correlates field numbers with positions on the screen and the field's contents.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### **fieldNumber**

The identifier of the field to move the cursor to the start of.

All of the operation's results except the following are described in [Common results in the integration](#).

## cursorRow

The current cursor row (zero-origin).

## cursorColumn

The current cursor column (zero-origin).

## Read Buffer

The **Read Buffer** operation dumps the contents of the screen buffer, one line at a time. The format of the output depends on the value you specify for the input **captureType**. The output format of this operation is a *<screen width>* by *<screen height>* table of hexadecimal codes, one for each character position on the screen. If you specify a value of **ASCII** for the **captureType** input, each hexadecimal code is the ASCII code for the character; if you specify a value of **EBCDIC**, each hexadecimal value is EBCDIC for each character.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### captureType

The format for the output data. The valid values are **ASCII** and **EBCDIC**. The default is **ASCII**.

All of the operation's results except the following are described in [Common results in the integration](#).

### data

The table of hexadecimal codes.

## Special characters

- If you specify a value of **ASCII** for the **captureType** input, the operation dumps the contents of the screen buffer, one line at a time. Positions inside data fields are generally output as 2-digit hexadecimal codes in the current display character set. If the current locale specifies UTF-8 (or certain DBCS character sets), some positions may be output as multi-byte strings (4-, 6-, or 8-digit codes). DBCS characters take two positions in the screen buffer: the first location is output as a multi-byte string in the current locale codeset, and the second location is output as a dash.

The start-of-field characters (each of which takes up a display position) are output as *SF(aa=nn[,...])*, where *aa* is a field attribute type and *nn* is its value.

Attribute	Values
c0 basic 3270	20 protected 10 numeric 04 detectable 08 intensified 0c non-display 01 modified
41 highlighting	f1 blink f2 reverse f4 underscore f8 intensify

Attribute	Values
42 foreground	f0 neutral black f1 blue f2 red f3 pink f4 green f5 turquoise f6 yellow f7 neutral white f8 black f9 deep blue fa orange fb purple fc pale green fd pale turquoise fe grey ff white
43 character set	f0 default f1 APL f8 DBCS

Extended attributes (which do not take up display positions) are output as *SA(aa=nn)*, with *aa* and *nn* having the same definitions as above (though the basic 3270 attribute will never appear as an extended attribute).

In addition, NULL characters in the screen buffer are reported as ASCII character 00 instead of 20, even though they should be displayed as blanks.

- If you specify a value of **EBCDIC** for the **captureType** input, it is equivalent to `Snap(Ascii)`, but with the data fields output as hexadecimal EBCDIC codes. Additionally, if a buffer position has the Graphic Escape attribute, it is displayed as *GE(xx)*.

## Send Characters

The **Send Characters** operation sends a string of characters to the host using the s3270 **String** action. See below for details on special characters which may be used with this operation.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### **characterString**

The string of characters to send to the s3270 application.

All of the operation's results are described in [Common results in the integration](#).

## Special characters

The simplest method for nested scripts is provided via the **String** action. The arguments to **String** are one or more double-quoted strings which are inserted directly as if typed. The C backslash conventions are honored as shown below. (Entries marked with an asterisk (\*) mean that after sending the AID code to the host, s3270 waits for the host to unlock the keyboard before further processing the string.)

\b	Left
\exxxx	EBCDIC character in hex
\f	Clear*
\n	Enter*
\pan	PA(n)*
\pfnn PF(nn)*	
\r	Newline
\t	Tab
\T	BackTab
\uxxxx	Unicode character in hex
\xxxxx	Unicode character in hex

The numeric values for the **\e**, **\u**, and **\x** sequences can be abbreviated to 2 digits. EBCDIC codes greater than 255 and some Unicode character codes represent DBCS characters, which will work only if s3270 is built with DBCS support and the host allows DBCS input in the current field.

**Note:** The strings are in ASCII and are converted to EBCDIC, so take care not to insert control codes.

There is also an alternate form of the **String** action called **HexString**, which is used to enter non-printing data. The argument to **HexString** is a string of hexadecimal digits, two per character. A leading 0x or 0X is optional. In 3270 mode, the hexadecimal data represent EBCDIC characters, which are entered into the current field. In NVT mode, the hexadecimal data represent ASCII characters, which are sent directly to the host.

In order to use **HexString**, you must use the **Execute Command** operation.

## Send PA Key

The **Send PA Key** operation sends the specified PA key to the host.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### paKey

The PA key number to send. The valid values are 1-3.

All of the operation's results are described in [Common results in the integration](#).

## Send PF Key

The **Send PF Key** operation sends the specified PF key to the host.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### paKey

The PF key number to send. The valid values are 1-24.

All of the operation's results are described in [Common results in the integration](#).

## Send SysReq

The **Send SysReq** operation sends the System Request key to the host.

All of the operation's inputs are described in [Common inputs in the integration](#).

All of the operation's results are described in [Common results in the integration](#).

## Snap

The **Snap** operation takes a snapshot of the screen data and stores it in memory, or manipulates a snapshot which was previously saved.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

### snapCommand

The command to perform on the snapshot data. The valid values are **Ascii**, **Cols**, **Ebcdic**, **ReadBuffer**, **Rows**, **Save**, **Status**, and **Wait**. The default is **Save**. See [Snap commands](#) below for more details.

### snapTimeout

An optional timeout for the **Wait** value of the **snapCommand** input. The default is to wait forever, which means that the operation fails after the time period specified in the **timeout** input to the **Connect** operation.

All of the operation's results except the following are described in [Common results in the integration](#).

### data

The data returned by the **snap** command.

## Snap commands

### Snap

Equivalent to **Snap(Save)** command.

### Snap(Ascii)

Performs the **Ascii** action on the saved screen image. (It can only retrieve a full screen ASCII dump of the snapshot screen.)

### Snap(Cols)

Returns the number of columns in the saved screen image.

### Snap(Ebcdic)

Performs the **Ebcdic** action on the saved screen image. (It can only retrieve a full screen.)

### **Snap(ReadBuffer)**

Performs the **ReadBuffer** action on the saved screen image.

### **Snap(Rows)**

Returns the number of rows in the saved screen image.

### **Snap(Save)**

Saves a copy of the screen image and its status in a temporary buffer. You can query this copy with other **Snap** actions to allow a script to examine a consistent screen image, even when the host may be changing the image (or even the screen dimensions) dynamically.

### **Snap(Status)**

Returns the status line from when the screen was last saved.

### **Snap(Wait[,timeout],Output)**

Pauses the script until the host sends further output, then updates the snap buffer with the new screen contents. This command is used when the host unlocks the keyboard (allowing the script to proceed after an **Enter**, **PF**, or **PA** action), but has not finished updating the screen. This action is usually invoked in a loop that uses the **Snap(Ascii)** or **Snap(Ebcdic)** action to scan the screen for some pattern that indicates that the host has fully processed the last command. The optional **timeout** parameter specifies a number of seconds to wait before failing the **Snap** action. The default is to wait indefinitely, which results in the operation failing after the **timeout** period specified in the **Connect** operation.

## **Wait**

The **Wait** operation waits for an event to occur, with an optional timeout. For a detailed explanation of the different events, and what the value of timeout will do, see the [Wait commands](#) section below.

All of the operation's inputs except the following are described in [Common inputs in the integration](#).

#### **waitCommand**

The event to wait for. The valid values are **3270Mode**, **Disconnect**, **InputField**, **NVTMode**, **Output**, **Unlock**, and **Seconds**. See [Wait commands](#) below.

#### **waitTimeout**

The timeout in seconds for the **Wait** command. This is optional for all commands except **Seconds**.

All of the operation's results except the following are described in [Common results in the integration](#).

#### **data**

The data returned by the **Snap** command.

## **Wait commands**

### **Wait([timeout,] 3270Mode)**

This command is used when communicating with a host that switches between NVT mode and 3270 mode. It pauses the script or macro until the host negotiates 3270 mode, then waits for a formatted screen.

The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

For backwards compatibility, **Wait(3270)** is equivalent to **Wait(3270Mode)**.

### **Wait([timeout,] Disconnect)**

This command pauses the script until the host disconnects. It is often used after sending a **logoff** command to a VM/CMS host to ensure that the session is not unintentionally set to the **disconnected** state. The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

### **Wait([timeout,] InputField)**

This command is a useful utility to use at the beginning of scripts and after the **Connect** action. In 3270 mode, it waits until the screen is formatted and the host has positioned the cursor on a modifiable field. In NVT mode, it waits until the host sends at least one byte of data. The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

### **Wait([timeout,] NVTMode)**

This command is used when communicating with a host that switches between 3270 mode and NVT mode. It pauses the script or macro until the host negotiates NVT mode, then waits for a byte from the host as above. The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

For backwards compatibility, **Wait(ansi)** is equivalent to **Wait(NVTMode)**.

### **Wait([timeout,] Output)**

This command pauses the script until the host sends further output. It is often needed when the host unlocks the keyboard (allowing the script to proceed after a **Clear**, **Enter**, **PF**, or **PA** action), but has not finished updating the screen. Also used in non-blocking AID mode (see DIFFERENCES at <http://x3270.bgp.nu/x3270-script.html#Differences> for details). This action is usually invoked in a loop that uses the **Ascii** or **Ebcdic** action to scan the screen for some pattern that indicates that the host has fully processed the last command.

The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

### **Wait([timeout,] Unlock)**

This command pauses the script until the host unlocks the keyboard. This is useful when operating in non-blocking AID mode (toggle AidWait clear), to wait for a host command to complete. See DIFFERENCES at <http://x3270.bgp.nu/x3270-script.html#Differences> for details). The optional **timeout** parameter specifies the number of seconds to wait before failing the **Wait** action. The default is to wait indefinitely.

### **Wait(timeout, Seconds)**

This command delays the script **timeout** seconds. Unlike the other forms of **Wait**, the **timeout** is not optional.

## **Troubleshooting**

This section provides troubleshooting procedures and tools you can use to solve problems you may encounter while using the TN3270 operations.

## General troubleshooting procedures and tools

### Connect to the host with a terminal emulator

Since the TN3270 operations use the s3270 program from your SSH relay host, it is usually best to debug things from that host. From the SSH relay host you can use the x3270 program if you prefer a graphical user interface, or c3270 if you do not have X windows available on the SSH relay host.

### Usage of the Connect operation

When the **Connect** operation is added to a flow, you must add a result to the **Connect** step of the flow. This result must be named **session** and be assigned from the step result **session**. If this result is not added to the flow step, the subsequent operations will fail with a message that states the session could not be retrieved from the context.

## Security

TN3270 operations use two connections which require security considerations.

### RAS to SSH relay host

The connection from your RAS server to the SSH relay host is through SSH, which provides an encrypted channel for all communications. Options are also available for public key authentication with the SSH relay host. For more information, see the **Connect** operation.

### SSH relay host to TN3270 host

The connection from the SSH relay host to the TN3270 host can either be unencrypted Telnet or Telnet tunneled through an encrypted SSL tunnel or the encrypted TN3270E protocol. Client authentication can also be done using certificate files on newer versions of s3270. For more information on this please see the details for the **Connect** operation, specifically for the **host** input.

## Tools

Following are OO tools that you can use with the TN3270 integration:

- **RSFlowInvoke.exe** and **JRSFlowInvoke.jar**

RSFlowInvoke (RSFlowInvoke.exe or the Java version, JRSFlowInvoke.jar) is a command-line utility that allows you to start a flow without using Central (although the Central service must be running). RSFlowInvoke is useful when you want to start a flow from an external system, such as a monitoring application that can use a command line to start a flow.



- **Web Services Wizard (wswizard.exe)**

When you run the Web Services Wizard, you provide it with the WSDL for a given Web service. The WSDL string you provide as a pointer can be a file's location and name or a URL. The Web Services Wizard displays a list of the methods in the API of the Web service that you specify. When you run the wizard, pick the methods you want to use, and with one click for each method you have selected, the wizard creates an HP OO operation that can execute the method. This allows you to use the Web Services Wizard to create operations from your monitoring tool's API.

These tools are available in the Operations Orchestration home folder in /Studio/tools/.