RSA NETWITNESS® PLATFORM

Shell User Guide

for RSA NetWitness Platform 11.3

Contact Information

RSA Link at https://community.rsa.com contains a knowledgebase that answers common questions and provides solutions to known problems, product documentation, community discussions, and case management.

Trademarks

For a list of RSA trademarks, go to www.emc.com/legal/emc-corporation-trademarks.htm#rsa.

License Agreement

Third-Party Licenses

This product may include software developed by parties other than RSA. The text of the license agreements applicable to third-party software in this product may be viewed on the product documentation page on RSA Link. By using this product, a user of this product agrees to be fully bound by terms of the license agreements.

Note on Encryption Technologies

This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when using, importing or exporting this product.

Distribution

Dell believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

February 2019

Table of Contents

Shell

This guide describes the shell utility `nw-shell` that can be used to troubleshoot the operations of RSA NetWitness Platform management services like security-server, investigate-server, correlation-server, and so on. It is the equivalent of the `NwConsole` utility used to interact with NetWitness Platform capture services like Decoders and Concentrators. This utility is independent of the business logic of the service, and works the same with most NetWitness Platform services.

Features

The `nw-shell` utility provides following features:

1. Supports secure connections to the local NetWitness Platform service instances.
2. Supports navigation of the service tree to explore its operational state.
3. Exposes current run state of the service through configuration, metrics, health-checks, and so on to help troubleshooting.
4. Supports scripting to automate simple administration tasks in field deployments.
5. Runs on Linux, OSX, and Windows terminals.

Installation

The `nw-shell` can be installed with the `rsa-nw-shell` RPM:

```
$ sudo yum install rsa-nw-shell
$ /usr/local/bin/nw-shell
RSA NetWitness Shell. Version: 4.0.0
See "help" to list available commands, "help connect" to get started.
offline »
```

Usage

The primary goal of the shell is to help a human operator explore the runtime state of a NetWitness Platform service. It is, essentially, an interactive program that invokes APIs on running NetWitness Platform services. Each service includes a system API that exposes its runtime state as a *logical* tree. The shell leverages the Tree API structure to present a hierarchical view of the service that is similar to a filesystem view. Users can navigate the tree using `cd` into directories to view or modify the corresponding configuration at the location, or invoke API methods.

The set of commands available to users at a given time depends on the current shell *context*, that is, their placement inside the *logical* tree. Certain commands, however, are always available, and we begin with a description of those.

System Commands

Shell *system* commands operate on the shell itself, instead of the nodes of a connected service. These commands are always available:

Command	Function	Example
`help [<command>]`	Provide help on available commands or a particular command	`help` or `help connect`
`clear`	Clear the screen (shortcut Ctrl-l)	`clear`
`exit` or `quit`	Exit the shell	`quit`
`history`	Display the history of previously run commands	`history`

`help <command>` is always available. Use it to explore the available commands.

`nw-shell` supports tab completion of the command, and the applicable parameter names wherever possible, for example, pressing the Tab key after typing in `e` completes the command to `exit`.

Shell also supports a non-interactive mode where it executes script from the provided file, used by specifying the absolute filepath prefixed by the `@` argument. See scripting for details.

```
AVAILABLE COMMANDS

Built-In Commands
        clear: Clear the shell screen.
        exit, quit: Exit the shell.
        help: Display help about available commands.
        history: Display or save the history of previously run commands
        script: Read and execute commands from a file.
        stacktrace: Display the full stacktrace of the last error.

Context Commands
      * cd: Change the current node. Usage: cd <path>
        connect: Connect to a service. One of --service or --port must be specified.Usage: connect [--service <service>[.<id>]] [--broker amqp://localhost/rsa/system] [--host
localhost] [--port] [--insecure false]
      * where: Which service shell is connected to?

Token Commands
        login: Authenticate to a service. Usage: login [connect-parameters]
        login-insecure: Authenticate to a service providing user and password on the command prompt. The password is recorded in the shell history so this command must be used with
care.Usage: login-insecure --user <user> --password <password> [connect-parameters]
      * logout: Clear the authentication context: logout
      * whoami: Who am I?

Tree Node Commands
      * json: Print the current node as a JSON string
      * show: Pretty print the current node

Tree Node List Commands
      * config: Summarize configuration of the current subtree
      * health: Summarize health of the current subtree
      * ls: List the children of the current node. Usage: ls [<filter>] [--values] [--types]
      * lsv: Shorthand for ls --values. Usage: lsv [<filter>] [--types]
      * method: Summarize methods of the current subtree
      * metrics: Summarize metrics of the current subtree
      * snapshot: Snapshot the current subtree

Tree Node Method Commands
      * invoke: Invokes the method that exists on the current method type node. Usage: invoke [argument] [--file jsonfile]

Tree Node Value Commands
      * get: Get the value of the current node
      * set: Set the value of the current node. Usage: set <new-value>

Commands marked with (*) are currently unavailable.
Type `help <command>` to learn more.
```

Help, History Commands Usage

```
offline » help connect
    connect - Connect to a service. One of --service or --port must be specified.Usage: connect [--service <service>[.<id>]] [--broker amqp://localhost/rsa/system] [--host
localhost] [--port] [--insecure false]
    ...
    ...
offline » history
help
help connect
history
offline »
```

Previously typed commands in shell can be accessed using the up and down arrow keys. By default, command history is written to the log file `/usr/sbin/nw-shell.log`. If needed, saving commands to the log file option can be turned off by setting the environment variable `RSA_SHELL_HISTORY_LOG` to `false`.

Authentication Commands

As an administration and monitoring tool, it is important that `nw-shell` authenticates users before handing them control over a running service. The following commands manage the shell authentication context:

Command	Function
`login`	Authenticate and retain tokens for future interactions.
`logout`	Clear the security context and token.
`whoami`	Print a summary of the current identity.

The `login` command can be used to authenticate the user and establish an identity. Service operators can login once (against the NetWitness Security Server) and then use the token to connect to multiple services and perform administration based on the roles assigned. This single-sign-on workflow is simplified by separating the `login` and `connect` commands to allow the operator to authenticate once (using `login`) and then switch services seamlessly using `connect`.

In general, the `login` command takes the same parameters as `connect` (described below) to specify the service that does the credential validation.

```
offline » login
user: admin
password: ********
admin@offline » connect --service respond-server
INFO: Connected to respond-server (23e1dab7-0658-41a5-bb1e-d716a37d5ea5)
admin@respond-server:Folder:/rsa »
admin@respond-server:Folder:/rsa » connect --service investigate-server
INFO: Connected to investigate-server 2f21db20-4b50-48cf-8f7f-c0be0d1d1d12)
admin@investigate-server:Folder:/rsa »
```

The current logged-in identity can be confirmed at any point using the `whoami` command.

```
admin@offline » whoami
+-------------+-------------------------------------------------+
|   Subject   |                      admin                      |
+-------------+-------------------------------------------------+
|    Issued By|security-server-407ca159-ccf9-4857-a1e8-8c0b1ce8d|
|    Issued At|2019-01-29T15:55:57.606Z                         |
|Expires After|2019-01-29T23:55:57.606Z                         |
|       Roles|[407ca159-ccf9-4857-a1e8-8c0982b1ce8d]           |
+-------------+-------------------------------------------------+
```

The shell security context can be cleared by using the `logout` command.

```
security-server:Folder:/rsa » logout
security-server:Folder:/rsa » whoami
You are not logged in.
```

Some operations, such as setting certain configuration properties or invoking a method, require a specific RBAC. To perform such privileged operations, you need to authenticate with an identity that has the necessary permissions.

Example:

```
security-server:Configuration:/rsa/logging/operations/max-file-count » get
10
security-server:Configuration:/rsa/logging/operations/max-file-count » set 15
ERROR: Failed to set the node value: Access is denied
security-server:Configuration:/rsa/logging/operations/max-file-count » login
user: admin
password: **********
admin@security-server:Configuration:/rsa/logging/operations/max-file-count » set 15
security-server:Configuration:/rsa/logging/operations/max-file-count » get
15
```

Context Changing Commands

The following commands change the current shell *context*:

Command	Function	Options	Examples
`connect`	Connect to a service	`service` `host`, `port`, `http`, `broker`	`connect --service security-server.d6a55b48⋯`
`cd`	Change context to a node		`cd /rsa/security`, `cd ../sys`, `cd ..`, `cd`

Connecting to a Service

The shell can connect to services over AMQP or HTTP(S), however AMP is the preferred and default option.

To connect to a specific instance of a service, the name and serviceId needs to be supplied in the format `connect --service {service-name}.{serivceId}`.
Example: `respond-server.d6a55b48-6103-46bd-9ead-3b4d589b302b`

If the service identifier is skipped, it is assumed to be `any`, that is `connect --service foo-server` will connect with any service named `foo-server` connected to the AMQP Broker.

Change Node

The `cd` command can be used to change the current node. Just like `cd` on a file system shell, it takes relative or absolute paths as input and changes the current context to the node at that path.

```
offline » cd
Command 'cd' was found but is not currently available because you are not connected to any service.

offline » connect
security-server:Folder:/rsa » cd log
security-server:Folder:/rsa/logging » cd ../security
security-server:Component:/rsa/security » cd /rsa/security/fips-mode
security-server:Configuration:/rsa/security/fips-mode » cd
security-server:Folder:/rsa »
```

The shell prompt summarizes and presents the user's current context. It starts off with `offline` and once the shell is connected to a service, it displays the service name it is connected to, and the type and the path of the current node. Once the user is authenticated, the userId is included in the prompt.

Invoking `cd` before connecting to a service reminds the user that some commands work only in certain contexts.

Some commands are available only in certain contexts. For example, `cd` works only when the shell is *online*. The `Help` command lists all the commands, but commands marked with (*) are unavailable in the given context.

Certain node-specific commands are enabled only when the current node is of a certain type. For example, method nodes support the command `invoke` which is not enabled for any other node type.

Node Display Commands

The following commands are available with all node types. They do not take any arguments and display the node details for the user to review.

Command	Function
`show`	Pretty print the current node.
`json`	Print the current node as a JSON string.

```
security-server:Configuration:/rsa/security/authentication/token-lifetime » show
+------------+-----------------------------------------+
|Configuration|/rsa/security/authentication/token-lifetime|
+------------+-----------------------------------------+
|        value|8 HOURS                                  |
|    valueType|com.rsa.asoc.launch.api.helpers.Seconds  |
| defaultValue|8                                        |
|  description|The time-to-live on a token.             |
+------------+-----------------------------------------+

security-server:Configuration » /rsa/security/authentication/token-lifetime » json
{
  "path" : "/rsa/security/authentication/token-lifetime",
  "type" : "Configuration",
  "value" : "10 HOURS",
  "parent" : {
    "path" : "/rsa/security/authentication",
    "type" : "Component"
  },
  "attributes" : {
    "defaultValue" : 10,
    "valueType" : "com.rsa.asoc.launch.api.helpers.Seconds",
    "description" : "The time-to-live on a token."
  }
}
```

The output of `json` dumps the API payload and may contain more details than those shown by `show`.

Node Value Commands

`Configuration`, `Metrics` and `Health` nodes have values. Their current values can be obtained using the `get` command.

Command	Function	Examples
`get`	Get the current value of the node.	
`set`	Update the value of the node.	`set false`, `set '{"enabled": false}'`

```
security-server:Gauge:/rsa/process/hostname » get
"hostxyz.corp.myorg.com"
security-server:Gauge:/rsa/process/hostname » cd /rsa/transport/http/secure
security-server:Configuration:/rsa/transport/http/secure » get
true
```

The value of a `Configuration` node can be changed by invoking the `set` command on it.

```
admin@security-server:Configuration:/rsa/security/pki/tls-protocols » show
+-------------+-------------------------------------------------------------------------------------+
|Configuration|                          /rsa/security/pki/tls-protocols                            |
+-------------+-------------------------------------------------------------------------------------+
|        value|[TLSv1.2]                                                                            |
|    valueType|[Ljava.lang.String;                                                                  |
| defaultValue|null                                                                                 |
|    processor|[{message=Change requires restart, ofComponent=[javax.net.ssl.SSLContext],           |
|             |type=com.rsa.asoc.launch.api.annotation.RequiresRestart}]                            |
|  description|This property controls the TLS protocol versions supported by the applications.      |
+-------------+-------------------------------------------------------------------------------------+

admin@ssecurity-server:Configuration:/rsa/security/pki/tls-protocols » set '["SSLv3"]'
admin@ssecurity-server:Configuration:/rsa/security/pki/tls-protocols » value
[
  "SSLv3"
]
```

Node List Commands

`Folder`, `Component` and `Method` nodes can contain other nodes as their children. The `ls` command can be used to list the children nodes, their types, and where available, their current values.

Command	Function	Options	Examples
`ls <filter>`	List the child nodes of the current node.	`types`, `values`	`ls`, `ls cfg`, `ls --values`
`lsv <filter>`	Shorthand for `ls <filter> --values`	`types`	`lsv metrics` `lsv name:fips`

```
security-server:Component:/rsa/process » ls
jvm                      Component
modules                  Component
current-time-utc         Gauge
current-time-utc-pretty  Gauge
fips140-mode             Gauge
hostname                 Gauge
mode                     Gauge
service-id               Gauge
service-name             Gauge
status                   Gauge
uptime                   Gauge
uptime-pretty            Gauge
version                  Gauge
version-full             Gauge
version-raw              Gauge
ready                    Method
shutdown                 Method

security-server:Component:/rsa/process » ls comp
jvm      Component
modules  Component

security-server:Component:/rsa/process » ls name:uptime
uptime         Gauge
uptime-pretty  Gauge

security-server:Component:/rsa/process » ls name:uptime --values
uptime         Gauge  3713854
uptime-pretty  Gauge  1 hour 1 minute 53 seconds

security-server:Component:/rsa/process » lsv
jvm
modules
current-time-utc         1483978142782
current-time-utc-pretty  2017-01-09T16:09:02.782Z
fips140-mode             true
hostname                 hostxyz.corp.myorg.com
mode                     Normal
service-id               1fb7572a-4d87-497e-a4da-802819c10a72
service-name             no-op-server
status                   Running
uptime                   512064
uptime-pretty            8 minutes 32 seconds
version                  0.0
version-full             0.0.0.0
version-raw              0.0.0.0
ready
shutdown
```

Listing commands are not available on nodes that do not have children (for example, configuration or metric nodes).

```
security-server:Component:/rsa/process » cd hostname
security-server:Gauge:/rsa/process/hostname » ls
Command 'ls' exists but is not currently available because you are not logged in.
```

Values displayed in `ls` and `lsv` may be truncated for presentation purposes. Use `get` on the value node to get the complete (unaltered) value of a node.

Method Node Commands

`method` command shows all the available methods under the current node and all the sub-folders under the current node.

```
security-server:Component:/rsa/process » method
/rsa/process/ready
/rsa/process/shutdown
```

On a node that is of `Method` type , the `invoke` command will invoke the method.

```
security-server:Method:/rsa/process/ready » invoke
{
  "ready" : true,
  "serviceId" : "84f36740-5ae7-409f-b14f-b17e98703983",
  "marketingVersion" : "0.0"
}
```

If the method signature takes `input`, then `invoke <input>` will invoke the method with given parameters.

```
security-server:Method:/rsa/health/get » show

+------+-----------------------------------------------------------------+
|Method|                        /rsa/health/get                          |
+------+-----------------------------------------------------------------+
|output|java.util.List<com.rsa.asoc.launch.api.health.api.HealthStatus>|
| input|java.lang.String                                                 |
+------+-----------------------------------------------------------------+

 ┌──────────────┬─────────┐
 │Metric  │Value│
 ├──────────────┼─────────┤
 │ failed │0    │
 │invoked │0    │
 │  timer │0.0  │
 └──────────────┴─────────┘

Method:/rsa/health/get » invoke 'rsa.health.checks.security-pki'
[
  {
    "name" : "rsa.security.pki.pki-health",
    "status" : "Unhealthy",
    "details" : {
      "Reason" : "Using a self-signed certificate"
    }
  }
]
```

For inline method execution with input payload, in cases where input contains any special characters such as `\`, `\n`, `\r`, `\t` , `"`, input needs to be properly escaped with a `\` .

In cases where method takes a complex payload, payload can be supplied with a file reference. In this case, payload does not need any special escaping.

```
Method:/rsa/configuration/collections/register» invoke --file /tmp/jsonfile.txt
```

Run State of a Service

Once shell is connected to a service, commands like `health`, `metrics`, `config` or `snapshot` can be used to retrieve the current state of the service at the current node and all the sub-folders under the current node.

Health

`health` command lists the health of the components in the current node and all the sub-folders under the current node.

```
admin@security-server:Component:/rsa/process » health
/rsa/process/jvm/memory-health      Healthy
/rsa/process/modules/module-health  Healthy
```

Metrics

`metrics` command lists all of the metrics available in the current node and all the sub-folders under the current node.

```
admin@respond-server:Component:/rsa/tree/node » metrics
/rsa/tree/node/get/invoked   14
/rsa/tree/node/get/timer     5666541.077918259
/rsa/tree/node/list/invoked  12
/rsa/tree/node/list/timer    6652400.379371729
```

In the above example, `/rsa/tree/node/get/invoked 14` shows that the method `get` is invoked `14` times and `timer` shows that it took on average `5666541.077918259` nano seconds to process the `get` request.

`cd` into the `timer` node shows more details like rate of requests for the current node for per minute and five minutes.

```
admin@respond-server:Timer:/rsa/tree/node/get/timer » show
+-----------------+------------------------+
|      Timer      |/rsa/tree/node/get/timer|
+-----------------+------------------------+
|            value|3267513.7521543643      |
|        valueType|java.lang.Double        |
|     stdDeviation|4381863.808014679       |
|   fiveMinuteRate|0.018725581010540414    |
|              max|181326026               |
|            count|27                      |
|       sampleSize|27                      |
|    oneMinuteRate|0.038516144812437314    |
|              min|810240                  |
|fifteenMinuteRate|0.009425563630894371    |
|           median|2146170.0               |
|         meanRate|5.189274295248098E-4    |
|             mean|3267513.7521543643      |
+-----------------+------------------------+
```

Config

`config` command shows all the configuration properties in the current node and all the sub-folders under the current node.

```
admin@respond-server:Component:/rsa/logging » config
/rsa/logging/audit/max-file-count       10
/rsa/logging/audit/max-file-size        10 MB
/rsa/logging/forward/categories         [Audit]
/rsa/logging/forward/destination        SYSLOG_UDP
/rsa/logging/forward/enabled            true
/rsa/logging/forward/host               localhost
/rsa/logging/forward/port               50514
/rsa/logging/forward/secure             false
/rsa/logging/levels
/rsa/logging/operations/max-file-count  10
/rsa/logging/operations/max-file-size   10 MB
```

Snapshot

`snapshot` command combines `metrics`, `configuration`, and `health` of the current state of the service node and all the sub-folders under the current node. This command is very useful for troubleshooting purposes. When opening a support case, it is encouraged to take a `snapshot` dump at the root node `\rsa` of a service that might be having issues and attach it to the case.

```
admin@security-server:Component:/rsa/security » snapshot
/rsa/security/account/external/get-all/invoked      16
/rsa/security/account/external/get-all/timer        NaN
/rsa/security/account/external/get/failed           6
/rsa/security/account/external/get/invoked          6
/rsa/security/account/external/get/timer            NaN
/rsa/security/account/force-password-change/invoke  1
/rsa/security/account/force-password-change/timer   NaN
/rsa/security/account/get/invoked                   123
/rsa/security/account/get/timer                     902163.0
/rsa/security/account/password-policy/cannot-inclu  false
/rsa/security/account/password-policy/min-chars     8
/rsa/security/account/password-policy/min-lower-ch  0
/rsa/security/account/password-policy/min-non-lati  0
/rsa/security/account/password-policy/min-numeric-  0
/rsa/security/account/password-policy/min-special-  0

.....
```

Scripting

`nw-shell` supports non-interactive executions that can take commands from files.

```
> cat /tmp/foo.script
connect --service security-server
cd /rsa/security/pki/ciphers
get
> time nw-shell @/tmp/foo.script
INFO: Connected to security-server (52c7b92c-23d5-4b3e-9973-31d8b1b27ec4)
[
  "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
  "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA",
  "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
  "TLS_DHE_RSA_WITH_AES_128_CBC_SHA"
]
nw-shell    5.89s user 0.33s system 305% cpu 2.032 total
```

Shell also has a built-in `script` which also can be used to execute scripts in the shell once the shell is launched.

```
 offline » script --file /tmp/foo.script
 INFO: Connected to security-server (52c7b92c-23d5-4b3e-9973-31d8b1b27ec4)
 [
   "TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256",
   "TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256",
   "TLS_DHE_RSA_WITH_AES_128_GCM_SHA256",
   "TLS_RSA_WITH_AES_128_CBC_SHA256"
 ]
```

The shell is primarily meant to help a human with exploring the runtime state of a NetWitness Platform service. For most other purposes, it is usually best to use the service published APIs.

Advanced Customization

The following JVM system properties can be used to customize certain aspects of `nw-shell` presentation:

System Property	Controls	Default Value
`console.width`	The width of text	100
`console.colors`	The use of colors	true
`console.prompt`	The prompt	`%s%s%s »`
`timeout`	The API timeout	30s

These can be specified using the `JAVA_OPTS` environment variable.

```
> export JAVA_OPTS="-Dtimeout=100s -Dconsole.width=40 -Dconsole.prompt=%s%s%s>"
> export JAVA_OPTS="-Dconsole.colors=false"
> nw-shell
```

The API timeout is a time unit which can take typical time unit values like 30 MINUTES, 30 mins, 30 m, 5 seconds, 5 sec, 5s, 5, 8000 milliseconds, and 8000 ms (ms = milliseconds).

The value of `console.prompt` must contain placeholders (that is, `%s`) for three strings which are replaced, in order, the

node type, a separator (`:`), and the node path. Setting `console.colors` to `false` will not turn off *all* use of colors, it only disables colors used by `nw-shell` in its output. The underlying libraries that the program uses may still output some text in color.

Tree View

NetWitness Platform services, such as security-server, investigate-server, and so on, are a collection of components working together to implement the goals of the service. This section describes the support for exposing these components and their associated elements (that is configuration parameters, metrics, API methods, health checks, and so on) in a `tree-view`. The `tree-view` is a hierarchically nested arrangement of `tree-node` s, each of which corresponds to a distinct functional area of the service. The arrangement enables the end-user to navigate the nodes at run time and explore the current state of the service. See Shell for an interactive tool that uses the tree-view.

Features

The `tree-view` supports following features:

1. Supports encapsulation of functionally related elements under the same `tree-node`.
2. Supports hierarchical linkage of `tree-node` s to enable navigation from one node to another.

Implementation

The current implementation of the `tree-view` is described next.

Node Types

The `tree-view` is comprised of `tree-node` instances taken from the following types.

Type	Represents
`Component`	A service component associated configuration, such as metrics, health, API, and so on.
`Configuration`	A configuration parameter that controls some aspect of the service behavior (for example, `rsa.filesystem.prefix`).
`Metric`	A *quantitative* measure of some aspect of the component operations (for example, Requests Processed).
`Health`	A *qualitative* assessment of some aspect of the component (for example, "Using too much memory").
`Method`	An API that can be invoked remotely to interact with the component (for example, /rsa/security/pki/set-certificate).
`Parameter`	A payload parameter that is an input or an output from an API.
`Folder`	A logical grouping of nodes that are related to one another but are not a component (for example, the root `/rsa`).

Node Structure

All `tree-node`s have the same structure. In particular, they include the following elements:

Item	Information
`path`	The node path is a `/` separated path like `/rsa/security/pki/api-secure`
`type`	The type of the node, taken from the table above.
`attributes`	A map of string keys to attribute values. The available keys depend on the type of the node.
`value`	The value of this node. This can be null if the node has no associated value (or if it has a null value).
`children`	A list of child nodes, if the node has any.
`parent`	The parent node. This is only null for the root node.