RSA NETWITNESS® SUITE

Guía del usuario de la interfaz de la línea de comandos

para la versión 11.0

Copyright © 1994-2017 Dell Inc. or its subsidiaries. All Rights Reserved.

Información de contacto

RSA Link en https://community.rsa.com contiene una base de conocimientos que responde a las preguntas comunes y brinda soluciones para problemas conocidos, documentación de productos, análisis de la comunidad y administración de casos.

Marcas comerciales

Para obtener una lista de las marcas comerciales de RSA, visite mexico.emc.com/legal/emc-corporation-trademarks.htm#rsa (visite el sitio web de su país correspondiente).

Acuerdo de licencia

Este software y la documentación asociada son propiedad e información confidencial de EMC, se suministran bajo licencia, y pueden utilizarse y copiarse solamente de acuerdo con los términos de dicha licencia y con el aviso de copyright mencionado a continuación. No se puede suministrar a ninguna persona, ni poner a su disposición de cualquier otra manera, este software ni la documentación, o cualquier copia de estos elementos.

Este documento no constituye ninguna transferencia de titularidad ni propiedad del software, la documentación o cualquier derecho de propiedad intelectual. Cualquier uso o reproducción sin autorización de este software y de la documentación pueden estar sujetos a responsabilidad civil o penal.

Este software está sujeto a cambios sin aviso y no debe considerarse un compromiso asumido por EMC.

Licencias de otros fabricantes

Este producto puede incluir software que ha sido desarrollado por otros fabricantes. El texto de los acuerdos de licencia que se aplican al software de otros fabricantes en este producto puede encontrarse en la página de documentación del producto en RSA Link. Al usar este producto, el usuario acepta regirse totalmente por los términos de los acuerdos de licencia.

Nota sobre tecnologías de cifrado

Es posible que este producto contenga tecnologías de cifrado. Muchos países prohíben o limitan el uso, la importación o la exportación de las tecnologías de cifrado, y las regulaciones actuales de uso, importación y exportación deben cumplirse cuando se use, importe o exporte este producto.

Distribución

EMC considera que la información de esta publicación es precisa en el momento de su publicación. La información está sujeta a cambios sin previo aviso.

febrero 2018

Contenido

Acceder a NwConsole y a la ayuda

RSA NetWitness Console, también conocido como NwConsole, es una aplicación de terminal de varias plataformas que proporciona herramientas sólidas y acceso de línea de comandos a los servicios principales, como Decoder, Log Decoder, Concentrator, Broker y Archiver. Aunque la mayoría de los usuarios realiza sus tareas e investigaciones a través de la interfaz del usuario de NetWitness Suite, algunos usuarios avanzados, como los administradores y los desarrolladores, requieren acceso directo a los servicios sin pasar por la interfaz del usuario. NwConsole permite ingresar comandos desde la línea de comandos o ejecutar varios comandos desde un archivo.

En este tema se describe cómo acceder a NwConsole y ver la ayuda interna en NwConsole.

La consola de RSA Security Analytics, también conocida como NwConsole, incluye amplia información de ayuda. Puede acceder a esta ayuda en la línea de comandos de Security Analytics.

Requisitos previos

Todos los dispositivos de NetWitness Suite tienen instalada la aplicación NwConsole. También puede instalarla en Windows, Mac y CentOS para conectarse e interactuar con un servicio Core.

NwConsole está disponible en la línea de comandos de un dispositivo de NetWitness Suite. Si accede de forma remota a un dispositivo principal, debe tener instalada la aplicación RSA NetWitness Console en una máquina con Windows, Mac o CentOS. Para obtener el instalador de aplicaciones de RSA NetWitness Console, póngase en contacto con atención al cliente de RSA.

Acceder a NwConsole

Para ejecutar NwConsole desde la línea de comandos en un dispositivo de NetWitness Suite o en un emulador de terminal, en el indicador <$>, escriba `NwConsole` (Linux) o `nwconsole` (Windows). El comando real es `NwConsole`, pero Windows no distingue mayúsculas de minúsculas. RSA NetWitness Console se muestra como aparece en el siguiente ejemplo.

```
Last login: Thu Sep 24 14:00:42 on console
usxx<username>m1:~ <username>$ NwConsole
RSA NetWitness Suite Console 10.6.0.0.6105
Copyright 2001-2015, RSA Security Inc.  All Rights Reserved.

Type "help" for a list of commands or "man" for a list of manual pages.
>
```

Ver ayuda

NwConsole proporciona ayuda para comandos individuales como también para temas específicos.

Precaución: Para obtener la información más reciente, vea el comando y los temas de ayuda enNwConsole.

Ver una lista de comandos

Para ver una lista de comandos disponibles y sus descripciones, en el indicador (>), escriba `help`. En el siguiente ejemplo se muestra una lista de los comandos disponibles.

```
> help
Local commands:
    avro2nwd      - Convert AVRO files to NWD files
    avrodump      - Display schema and contents of AVRO file (for debugging)
    blockspeed    - Tests various write block sizes to determine best setting
    compileflex   - Compile all flex parsers in a directory
    createflex    - Create a flex parser that matches tokens read from a file
    dbcheck       - Perform a database integrity check over one or more
                    session, meta, packet, log or stat db files
    diskspeed     - Measures the speed of the disk(s) mounted at a specified
                    directory
    echo          - Echos the passed in text to the terminal
    encryptparser - Encrypt all parsers in a directory
    feed          - Create and work with feed files
    fmanip        - Manipulate a file with XOR and check for embedded PEs
    hash          - Creates or verifies hashes of database files
    help          - Provides help information for recognized console commands
    history       - Displays, erases or executes a command in the command
                    history
    httpAggStats  - Tests HTTP aggregation and reports statistics as it
                    continues
    log           - Perform operations on a log database
    logParse      - Parse line delimited logs on stdin and post results to
                    stdout
    logfake       - Create a fake log pcap file
    lua           - Execute a lua script
    makec3        - Generate C3 Test Data
```

```
    makepcap      - Convert packet database files to pcap or log files
    man           - Displays a list of topics or opens a specific manual page
                    on a topic
    metaspeed     - Tests read performance over an existing meta db
    netbytes      - Display statistics on network interface utilization
    nwdstrip      - Convert full NWD file into just session and meta file
    pause         - Wait for user input when running a script file
    reindex       - reindex a collection
    sdk           - Execute SDK commands based on the C SDK library, type "sdk
                    help" for more information
    sleep         - Sleeps for the specified milliseconds
    timeout       - Globally change the timeout for waiting for a response from
                    a service
    tlogin        - Open a trusted SSL connection to an existing service
    topQuery      - Returns the top N longest running queries from the audit
                    log (either a file or from the log API)
    vslice        - Validate index slices

Remote commands (executed on the connected service, see "login"):
    login         - Connect to a remote service.  Once connected, type help to
                    see commands available for remote execution.

For detailed help, type "help <command>"
>
```

Ver ayuda detallada sobre un comando

Para ver información detallada acerca de un comando, escriba `help <command>`. En el siguiente ejemplo se muestra la ayuda para el comando `logParse` después de que se escribe `help logParse`.

```
For detailed help, type "help <command>"
> help logParse
Usage: logParse {in=<pathname>} {indir=<pathname>} [out=<pathname>]
               [content=<c2|c3>] [device=<device,[device...]>]
               [path=<log-parsers-config-path>] [metaonly] [srcaddr=<src
               address>] [srcaddrfile=<filename,IP Address>]
Parse line delimited logs on stdin and post results to stdout

    in          - The input source file. "in=stdin" means interactive typing of
                  log.
```

```
    indir       - The input source files parent directory
    out         - The output file or output file parent directory if input is
                  set by indir. If not specified, use stdout as output.
    content     - Content version, either c2 or c3. Default is c2.
    device      - Comma delimited device list specifying devices that is
                  enabled. Default enable all devices.
    path        - The logparsers configuration path. Default will find
                  configuration file like logdecoder.
    metaonly    - The output will only contains parsed meta, otherwise will
                  print log message after metas.
    srcaddr     - The source address of the all the logs
    srcaddrfile - The source address for logs in one input file, in the format
                  filename,ipaddress
>
```

Ver una lista de temas de ayuda

Para ver una lista de temas de ayuda, escriba `man`. En el siguiente ejemplo se muestra una lista de temas de ayuda.

```
> man
List of topics:

    Introduction
    Connecting to a Service
    Monitoring Stats
    Feeds
    Converting Packet DB Files to PCAP
    Packets
    Verifying Database Hashes
    SDK Content
    SDK Content Examples
    Troubleshooting

Type "man <topic>" for help on a specific topic, partial matches are acceptable
>
```

Ver un tema de ayuda específico

Para ver ayuda sobre un tema específico, escriba `man <topic>`. En el siguiente ejemplo se muestra el tema de ayuda Paquetes después de que se escribe `man Packets`.

```
Type "man <topic>" for help on a specific topic, partial matches are acceptable
> man Packets
                                        Packets
                                        =======

The *packets* command can be used to generate a pcap or log file based on a
list of Session IDs, a time period or a where clause.  The command is quite
flexible and can be used on any running service that has access to the raw
data from a downstream component.  Before running the command, you must first
*login* to a service and then change directory to the appropriate sdk node,
(e.g., "cd /sdk").  Unlike the *makepcap* command, which only works on the
local file system, this command is meant to be used on a remote service.

    login ...
    cd /sdk
    packets where="service=80 && time='2015-03-01 15:00:00'-'2015-03-01
    15:10:00'" pathname="/tmp/march-1.pcap"
Write 10 minutes of HTTP only packets from March 1st, to the file
/tmp/march-1.pcap.  All times are in UTC.

    packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
    pathname=/media/sdd1/packets.pcap.gz
Write all packets between the two times to a gzip compressed file at
/media/sdd1/packets.pcap.gz

    packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
    pathname=/media/sdd1/mylogs.log
Write all logs between the two times to a plaintext file at
/media/sdd1/mylogs.log.  Any pathname ending with .log indicates that the
format of the output file should be plaintext line-delimited logs.
>
```

Precaución: Para obtener la información más reciente, vea el comando y los temas de ayuda enNwConsole.

Salir de NwConsole

Para salir de la aplicación NwConsole, escriba `quit` en la línea de comandos.

Parámetros básicos de la línea de comandos y edición

NwConsole es como una navaja suiza; hay todo tipo de herramientas ocultas debajo de su interfaz de línea de comandos. NwConsole es una plataforma múltiple; hay disponibles archivos ejecutables para CentOS (se envía en los dispositivos), Windows y Mac.

Parámetros básicos de la línea de comandos

Estos son algunos parámetros básicos de la línea de comandos:

- Para ejecutar un conjunto de comandos desde un archivo:
  `NwConsole -f /tmp/somefile.script`
- Para pasar una lista de comandos de la línea de comandos:
  `NwConsole -c <command1> -c <command2> -c <command3>`
  Esto no se recomienda necesariamente, excepto para scripts muy simples. El intérprete de Bash puede destruir las cadenas entre comillas si no sale correctamente. Si experimenta errores que no son evidentes al pasar a través de la línea de comandos, cambie para leer desde un archivo para ver si se solucionan los problemas.
- Por lo general, la consola sale después de ejecutar los comandos que se transmiten a través de un archivo o línea de comandos, pero si desea mantener abierto el indicador interactivo después de que se ejecutan los comandos, ingrese `-i` en la línea de comandos.
- Y, por supuesto, puede ejecutar NwConsole y escribir los comandos en la ventana de la consola.

Edición de líneas

Puede utilizar las claves de la tabla siguiente cuando edita un comando.

Clave	Descripción
Ctrl-U	Borra la línea actual
Ctrl-W	Elimina la palabra en la que se encuentra el cursor
Ctrl-A	Mueve el cursor al principio de la línea
Ctrl-E	Traslada el cursor hasta el final de la línea

Clave	Descripción
Flecha hacia arriba	Muestra el comando ejecutado con anterioridad
Flecha hacia abajo	Muestra el comando que se ejecuta después del comando actual (solamente es válido si se ha presionado la flecha hacia arriba)
Flecha hacia la izquierda	Mueve el cursor al carácter anterior
Flecha hacia la derecha	Mueve el cursor al carácter siguiente
Pestaña	Proporciona el completado contextual de la mayoría de los comandos y sus parámetros. La tecla de tabulación es muy útil para las tareas de edición. Por ejemplo, para ver el tema de ayuda *Conexión a un servicio*, puede escribir `mancon` en la línea de comandos y, a continuación, presionar la tecla `Tab`. NwConsole completa el comando por usted: `man Connecting to a Service` Presione Intro para ejecutar el comando y ver el tema.
`history`	Muestra una lista numerada de comandos anteriores
`history execute=#`	Ejecuta un comando anterior, lo cual también es equivalente a escribir `!#` Por ejemplo, `!1` ejecuta el comando anterior.
`history clear`	Borra el historial de todos los comandos
`history erase=#`	Borra un comando específico desde el búfer de historial. El historial se almacena automáticamente de una sesión a la siguiente.

Establecimiento de conexión con un servicio

Para conectarse e interactuar con un servicio Security Analytics Core (Decoder, Concentrator, Broker, Archiver, etc.), primero debe emitir el comando `login`. Debe tener una cuenta en ese servicio. Puede ingresar `type help` en cualquier momento para obtener más información. Esta es la sintaxis del comando `login`:

```
login <hostname>:<port>[:ssl] <username> [password]
```

Por ejemplo: **login 10.10.1.15:56005:ssl someuser**

Si no incluye la contraseña, se le solicita y se realiza el enmascaramiento de contraseña adecuado.

Si configuró confianza adecuada entre NwConsole y el terminal, puede usar el comando `tlogin` y evitar tener que ingresar una contraseña. La configuración de confianza va más allá del alcance de esta documentación, pero implica agregar certificados SSL de NwConsole al terminal mediante el comando `send /sys peerCert op=add --file-data=<pathname of cert>`. Primero debe usar un inicio de sesión normal con los permisos correctos para poder agregar un certificado de par para los inicios de sesión de confianza posteriores.

Una vez conectado, puede interactuar con el servicio del terminal mediante un sistema de archivos virtual. En lugar de archivos, lo que puede ver son los nodos de ese servicio. Algunos nodos son carpetas que tienen nodos secundarios, lo cual forma una estructura jerárquica. Cada nodo tiene un propósito y todos ellos son compatibles con un subconjunto de comandos como `info` y `help`. El mensaje `help` muestra información sobre los comandos compatibles con cada nodo. Cuando inicia sesión por primera vez, se encuentra en el nodo raíz, que es la ruta `/`, igual que en un sistema Mac o Linux. Para ver una lista de nodos que están debajo de `/`, ingrese el comando `ls`.

Todos los servicios tienen nodos como `sys` y `logs`. Para interactuar con la API **/logs**, primero puede enviar el comando `help` al nodo **/logs**. Para ello, debe usar el mensaje `send`, el cual tiene la siguiente sintaxis:

```
Usage: send {node pathname} {message name} [name=value [name=value]]

            [--file-data=<pathname>] [--string-data=<text>] [--binary-data=<text>]
            [--output-pathname=<pathname>] [--output-append-pathname=<pathname>]
            [--output-format={text,json,xml,html}]
Sends a command to a remote pathname.  For remote help, use "send <pathname>help"
for details.

    pathname                - The node pathname to retrieve information on
```

```
    message                  - The command (message) to send
    parameters               - Zero or more name=value parameters for the command
    --file-data              - Loads data from a file and send as either a BINARY
                               message or as a PARAMS_BINARY message if other
                               parameters exist
    --string-data            - Sends text as a STRING message type
    --binary-data            - Send text as either a BINARY message type or as a
                               PARAMS_BINARY message type if other parameters
                               exist
    --output-pathname        - Writes the response output to the given pathname,
                               overwriting any existing file
    --output-append-pathname - Writes the response output to the given pathname,
                               will append output to an existing file
    --output-format          - Writes the response in one of the given formats,
                               the default is text
```

Por lo tanto, para enviar un mensaje `help`, debe enviar esto:

```
send /logs help
```

Y su respuesta sería algo como esto:

```
    description: A container node for other node types
    security.roles: everyone,logs.manage
    message.list: The list of supported messages for this node
    ls:  [depth:<uint32>] [options:<string>] [exclude:<string>]
    mon:  [depth:<uint32>] [options:<uint32>]
    pull:  [id1:<uint64>] [id2:<uint64>] [count:<uint32>] [timeFormat:<string>]
    info:
    help:  [msg:<string>] [op:<string>] [format:<string>]
    count:
    stopMon:
    download:  [id1:<uint64>] [id2:<uint64>] [time1:<date-time>] [time2:<date-
time>] op:<string>
  [logTypes:<string>] [match:<string>] [regex:<string>] [timeFormat:<string>]
[batchSize:<uint32>]
    timeRoll:  [timeCalc:<string>] [minutes:<uint32>] [hours:<uint32>]
[days:<uint32>] [date:<string>]
```

Para obtener más información acerca de un comando o mensaje específicos, puede especificar `msg=<message name>` en el comando de ayuda como un parámetro. Por ejemplo, observe la ayuda del mensaje `pull`:

```
send /logs help msg=pull
```

```
    pull: Downloads N log entries
    security.roles: logs.manage
    parameters:
    id1 - <uint64, optional> The first log id number to retrieve, this is mutually
exclusive with id2
    id2 - <uint64, optional> The last log id number that will be sent, defaults to
most recent log
 message when id1 or id2 is not sent
    count - <uint32, optional, {range:1 to 10000}> The number of logs to pull
    timeFormat - <string, optional, {enum-one:posix|simple}> The time format used
in each log message,
 default is posix time (seconds since 1970)
```

La ayuda del mensaje integrada indica que este comando explora las últimas n entradas del registro si deja fuera id1 e id2. Para mirar las últimas 10 entradas del registro en este servicio:

```
send /logs pull count=10 timeFormat=simple
```

Casi todos los comandos en el servicio siguen este formato simple. Los únicos comandos que no lo siguen son aquellos que requieren un protocolo de enlace más complejo, como la importación de un PCAP a un Decoder. Para importar un PCAP, use el comando `import` de NwConsole, que se encarga del protocolo de enlace de canal de comunicación complejo.

Algunos parámetros son específicos del comando `send` de NwConsole y no se envían realmente al servicio. Puede utilizar estos parámetros para cambiar el formato de salida de la respuesta, escribir la respuesta en un archivo o leer un archivo desde la máquina local y enviarlo al servicio. Los parámetros locales para el comando `send` de NwConsole comienzan con dos guiones -.

- `--output-format`: Este parámetro cambia la salida normal del comando de texto sin formato a uno de estos tipos: JSON, XML o HTML.
- `--output-pathname`: En lugar de escribir la salida en el terminal, la escribe en el nombre de ruta especificado (trunca los archivos existentes).
- `--output-append-pathname`: Esto es igual a `--output-pathname`, salvo que anexa la salida a un archivo existente (o crea el archivo si no existe).
- `--file-data`: Lee un archivo y lo utiliza como la carga útil del comando. Esto es útil para comandos como `/sys fileEdit`. En el siguiente ejemplo se muestra cómo se puede enviar un archivo **index-concentrator-custom.xml** actualizado mediante NwConsole:

```
send /sys fileEdit op=put filename=index-concentrator-custom.xml --file-
data="/Users/user/Documents/index-concentrator-custom.xml"
```

- `--file-format`: Cuando se lee un archivo de entrada con `--file-data`, este parámetro obliga a NwConsole a interpretar el archivo como un tipo de entrada específico. Las enumeraciones permitidas son las siguientes: `binary`, `params`, `param-list`, `string` y `params-binary`. Por ejemplo, para enviar un archivo de reglas de aplicación (`*.nwr`) a un Decoder, puede usar este comando:

```
send /decoder/config/rules/application replace --file-
data=/path/rules.nwr --file-format=param-list
```

- `--string-data`: Envía la carga útil del comando como una cadena en lugar de una lista de parámetros.
- `--binary-data`: Envía la carga útil de comandos como archivo binario en lugar de una lista de parámetros.

Ejemplo de transmisión de consulta a un archivo JSON (podría ser un conjunto de resultados grande):

```
send /sdk query size=0 query="select * where service=80 && time='2015-
03-05 13:00:00'-'2015-03-05 13:59:59'" --output-format=json --output-
pathname=/tmp/query.json
```

Algo que se debe tener en cuenta acerca del comando send es el hecho de que, de forma predeterminada, el tiempo de espera de una respuesta es de 30 segundos. Algunos comandos (como la consulta anterior) pueden tardar más en recibir los resultados. Para evitar un tiempo de espera prematuro del lado del cliente, puede usar el comando `timeout [secs]` para aumentar la espera. Por ejemplo, `timeout 600` esperaría una respuesta durante 10 minutos antes de que se agote el tiempo de espera. Una vez que aplica, surte efecto para todos los comandos posteriores.

Para desplazarse por la jerarquía de nodo virtual del servicio, puede usar el comando `cd` como lo haría en cualquier shell de comandos. Esto abarca los aspectos básicos de la conexión y la interacción con un servicio. Una vez que está conectado, el comando `help` enumera todos los comandos que puede utilizar para interactuar con el terminal. Estos comandos no se muestran cuando no está conectado a un terminal.

Monitoreo de estadísticas

Puede usar NwConsole para ver en tiempo real el cambio de estadísticas en un servicio. Sin embargo, tenga en cuenta que esto puede generar una de salida considerable. Si no tiene cuidado y monitorea una cantidad excesiva de nodos, la pantalla se desplaza demasiado rápido dejando de ser útil.

Por ejemplo, si inicia sesión en un Decoder, puede monitorear la velocidad de captura en tiempo real. Para hacerlo, ejecute estos comandos después de conectarse a un Decoder:

```
decoder/stats
```

```
mon capture.rate
```

Eso es todo lo que debe hacer. Ahora, en cualquier momento en que ocurre un cambio en la velocidad de captura, la salida se produce en la ventana de la consola.

Puede agregar otro monitoreo:

```
mon capture.avg.size
```

Ahora inspecciona esas dos estadísticas y genera una salida de esos valores cuando cambian. Probablemente observó que cuando intentó escribir el segundo comando, la salida del monitoreo original confundió su visualización. Este es el problema del monitoreo de estadísticas. Realmente no está diseñado para hacer nada más que solo observar las estadísticas después de que se ingresa el primer comando.

Sin embargo, puede detener el monitoreo. Para esto, escriba `delmons` y presione **Intro**. Omita la salida mientras escribe y se devolverá a una línea de comandos adecuada. Si desea monitorear muchas estadísticas a la vez, solo debe dar la ruta de la carpeta de estadísticas primaria y se monitorearán todas las estadísticas debajo de ella. Por ejemplo, si escribe `mon /decoder/stats` o `mon .` (son equivalentes), se monitoreará todo. Esté preparado para recibir una gran cantidad de salida. Recuerde que debe ingresar `delmons` si se desplaza demasiado rápido.

Comandos útiles

Los siguientes comandos de NwConsole son útiles al interactuar con servicios de Servidor de NetWitness Core:

- **feed**: Permite crear y trabajar con archivos de feed.
- **makepcap**: Convierte los archivos de base de datos (DB) de paquete en PCAP.
- **packets**: Recupera registros o paquetes desde el servicio que tiene iniciada la sesión.
- **hash**: Crea o verifica hashes de archivos de base de datos.

Las siguientes secciones, así como las páginas de información de temas (man) y de ayuda de NwConsole, ofrecen información adicional.

Feeds

El comando `feed` proporciona varias utilidades para crear y examinar los archivos de feed. Un archivo de feed contiene la definición y los datos de un único feed en un formato que se ha precompilado para que puedan cargarse de forma eficiente mediante un Decoder o Log Decoder. Para obtener una referencia completa sobre las definiciones de feed, consulte **Archivo de definiciones de feed** en la *Guía de configuración de Decoder y Log Decoder*.

create

```
feed create <definitionfile> [-x <password>]
```

El comando `feed create` genera archivos de feed para cada feed definido en un archivo de definición de feed. Un archivo de definición es un documento XML que contiene una o más definiciones. Cada definición de feed especifica un archivo de datos y la estructura de ese archivo de datos. Los archivos de feed resultantes se crearán en el mismo directorio que el archivo de definición con el mismo nombre del archivo de datos, pero con la extensión modificada a **.feed** (por ejemplo, **datafile.csv** da como resultado **datafile.feed**). Se sobrescribirán todos los archivos existentes con el nombre objetivo sin indicador alguno.

```
$ ls
example-definition.xml    example-data.csv
$ NwConsole
RSA NetWitness Console 10.5.0.0.0
Copyright 2001-2015, RSA Security Inc.  All Rights Reserved.

Type "help" for a list of commands or "man" for a list of manual pages.
> feed create example-definition.xml
Creating feed Example Feed...
```

```
done.  2 entries, 0 invalid records
All feeds complete.
> quit
$ ls
example-definition.xml    example-data.feed    example-data.csv
$
```

De manera opcional, los archivos de feed se pueden ocultar con la opción `-x` seguida de una contraseña de un mínimo de 16 caracteres (sin espacios). Esto se aplicará a todos los feeds definidos en el archivo de definición. Además del archivo de feed, se generará un archivo de token para cada archivo de feed. El archivo de token se debe implementar con el archivo de feed correspondiente.

```
feed create example-definition.xml -x 0123456789abcdef
```

Estadísticas

```
feed stats <feedfile>
```

El comando `feed stats` proporciona información resumida correspondiente a un archivo de feed no oculto existente. La especificación de un archivo de feed oculto generará un error.

```
> feed stats example.feed
 Example Feed stats:
 version  : 0
 keys count  : 1
 values count: 2
 record count: 2
 meta key  : ip.src/ip.dst
 language keys:
 alert    Text
```

dump

```
feed dump <feedfile> <outfile>
```

El comando `feed dump` genera una lista de pares clave-valor normalizados de un archivo de feed no oculto. Puede usar el archivo resultante para validar un archivo de feed o ayudar a determinar qué registros se consideraron no válidos cuando se creó el feed. La especificación de un archivo de feed oculto generará un error. Si existe `outfile`, el comando anulará sin sobrescribir el archivo existente.

```
feed dump example.feed example-dump.txt
```

Conversión de archivos de base de datos de paquetes en PCAP

Puede usar el comando `makepcap` para convertir rápidamente cualquier archivo de base de datos de paquetes en un archivo PCAP genérico y mantener el orden de tiempo de captura. Este comando ofrece muchas opciones (consulte `help makepcap`), pero es fácil de usar. Para comenzar, solo se necesita es el directorio de base de datos de paquetes (a través del parámetro `source=<pathname>`).

Nota: Debe detener el servicio Decoder o Archiver antes de ejecutar este comando. Si desea generar una PCAP mientas se ejecuta el servicio, consulte el comando `packets`.

```
makepcap source=/var/netwitness/decoder/packetdb
```

Este comando convierte cada archivo de base de datos de paquetes en un archivo PCAP correspondiente en el mismo directorio. Si el disco está casi lleno, consulte el siguiente comando.

```
makepcap source=/var/netwitness/decoder/packetdb
dest=/media/usb/sde1
```

Este comando escribe todas las PCAP de salida en el directorio en **/media/usb/sde1**.

```
makepcap source=/var/netwitness/decoder/packetdb
dest=/media/usb/sde1 filenum=4-6
```

Este comando solo convierte los archivos numerados 4 a 6 y omite todos los demás. En otras palabras, convierte los archivos de base de datos de paquetes: **packet-000000004.nwpdb**, **packet-000000005.nwpdb**y **packet-000000006.nwpdb**.

```
makepcap source=/var/netwitness/decoder/packetdb time1="2015-03-
01 14:00:00" time2="2015-03-02 07:30:00" fileType=pcapng
```

Este comando solo extrae los paquetes con un registro de fecha y hora entre 1 de marzo de 2015 a las 14:00 h y 2 de marzo de 2015 a las 7:30 h o antes. Escribe el archivo como pcapng en el mismo directorio que el origen. Todos los registros de fecha y hora son UTC.

Paquetes

Puede usar el comando `packets` para generar un archivo PCAP o un archivo de registro según una lista de ID de sesión, un período o una cláusula Where. Este comando es muy flexible, se puede utilizar en cualquier servicio de ejecución que tenga acceso a los datos crudos de un componente descendente. Antes de ejecutar el comando, primero debe `login` en un servicio y, a continuación, cambiar de directorio al nodo sdk adecuado (por ejemplo, `cd /sdk`). A diferencia del comando `makepcap`, que solo funciona en el sistema de archivos local, use este comando para un servicio remoto.

```
login ...

cd /sdk
```

```
packets where="service=80 && time='2015-03-01 15:00:00'-'2015-03-01 15:10:00'"
pathname="/tmp/march-1.pcap"
```

Este comando escribe 10 minutos de paquetes solo HTTP desde el 1 de marzo en el archivo **/tmp/march-1.pcap**. Todas las horas son en UTC.

```
packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
pathname=/media/sdd1/packets.pcap.gz
```

Este comando escribe todos los paquetes entre las dos horas en un archivo comprimido GZIP en **/media/sdd1/packets.pcap.gz**.

```
packets time1="2015-04-01 12:30:00" time2="2015-04-01 12:35:00"
pathname=/media/sdd1/mylogs.log
```

Este comando escribe todos los registros entre las dos horas en un archivo de texto simple en **/media/sdd1/mylogs.log**. Cualquier nombre de ruta que termine con **.log** indica que el formato del archivo de salida debe ser un registro de texto simple delimitado por líneas.

Verificación de hashes de base de datos

De forma predeterminada, Archiver escribe un archivo XML por cada archivo de base de datos que se escribe. Este archivo XML finaliza con la extensión **.hash** y contiene un valor de hash del archivo junto con otra información pertinente. Puede usar el comando `hash` para verificar que el archivo de base de datos no se haya alterado mediante la lectura del hash almacenado en el archivo XML y, a continuación, la reaplicación de hash al archivo de base de datos para verificar que el hash sea válido.

```
hash op=verify
hashfile=/var/netwitness/archiver/database0/alldata/packetdb/pa
cket-000004880.nwpdb.hash
```

Este comando verifica que el archivo de base de datos de paquetes **packet-000004880.nwpdb** coincide con el hash del archivo XML **packet-000004880.nwpdb.hash**. Para la adecuada seguridad, el archivo hash se debe almacenar en otro lugar para impedir que altere el archivo XML (como escribir una vez únicamente medios), pero el comando hash en sí no distingue donde está almacenado.

Comando SDK Content

Uno de los comandos importantes en NwConsole es `sdk content`. Contiene varias opciones para hacer casi cualquier cosa, por lo menos en relación con la extracción de contenido de la plataforma NetWitness Suite Core. Puede usarlo para crear archivos PCAP, archivos de registro o extraer archivos de las sesiones de red (por ejemplo, explorar todas las imágenes de las sesiones de correo electrónico). Puede anexar archivos, tener un tamaño máximo asignado antes de crear un nuevo archivo y limpiar automáticamente los archivos cuando el tamaño del directorio aumenta demasiado. Se pueden ejecutar consultas en segundo plano para encontrar nuevas sesiones. Divide las consultas en grupos administrables y lleva a cabo automáticamente esas operaciones. Cuando se agota el grupo, hace una nueva consulta para obtener un nuevo conjunto de datos para otras operaciones. La lista de opciones del comando sdk content es muy amplia.

Debido a que el comando tiene muchas opciones, este documento proporciona ejemplos de comandos para diferentes casos de uso.

Antes de poder ejecutar `sdk content`, primero se deben ejecutar algunos comandos (como iniciar sesión en un servicio). Estos son algunos ejemplos:

- En primer lugar, conéctese a un servicio:
  `sdk open nw://admin:netwitness@10.10.25.50:50005`
- Si necesita conectarse a través de SSL, utilice el protocolo nws:
  `sdk open nws://admin:netwitness@10.10.25.50:56005`
- Tenga en cuenta que está pasando una dirección URL y debe aplicarle codificación de URL correctamente. Si la contraseña es `p@ssword`, la dirección URL se ve así: `sdk open nw://admin:p%40ssword@10.10.25.50:50005`
  Esto también se aplica al nombre de usuario.
- Una vez que inicia sesión, puede establecer un directorio de salida para los comandos: `sdk output <some pathname>`
- Para obtener ayuda de la línea de comandos, escriba: `sdk content`

Antes de probar algunos ejemplos de comandos, es importante comprender el parámetro `sessions`. Este parámetro es muy importante y controla la cantidad de datos que desea explorar (la cláusula where también es importante). El parámetro sessions es un ID de sesión único o un rango de ID de sesión. Todos los servicios de NetWitness Suite Core funcionan con ID de sesión, que comienzan en 1 y aumentan en 1 por cada sesión nueva que se agrega al servicio (sesión de red o registro). Los ID de sesión son números enteros de 64 bits, por lo que pueden llegar a ser muy grandes. Para hacerlo simple, supongamos que hay un Log Decoder que ha recopilado y analizado 1,000 registros. En el servicio, ahora tiene 1,000 sesiones con ID de sesión entre 1 y 1,000 (el ID de sesión 0 nunca es válido). Si desea que funcionen en las 1,000 sesiones, pase sesiones= 1-1000. Si solo desea que funcionen en las últimas 100 sesiones, pase sesiones= 901-1000. Una vez que el comando termina de procesar la sesión 1,000, regresa al indicador de la consola.

Sin embargo, muchas veces no nos interesan los rangos de sesiones específicos. Solo queremos ejecutar una consulta en todos ellos y procesar las sesiones que coinciden con la consulta. Estos son algunos accesos directos que simplifican este proceso:

- La letra `l` (L en minúscula) significa límite inferior o el ID de sesión más bajo.
- La letra `u` significa el ID de sesión más alto. De hecho, significa el ID de sesión más alto para sesiones futuras. En otras palabras, si pasa `sessions=l-u`, este rango especial significa operar en todas las sesiones actuales del sistema, pero también no salir del procesamiento y, a medida que ingresan nuevas sesiones al sistema, procesarlas también. El comando pausa y espera nuevas sesiones una vez que llega a la última sesión en el servicio. Para resumir, el comando nunca sale y entra en modo de procesamiento continuo. Se ejecuta durante días, meses o años, a menos que se interrumpa.
- Si no desea que el comando se ejecute para siempre, puede pasar `now` para el límite superior. Esto determina el ID de la última sesión en el servicio en el momento en que el comando se inicia y procesa todas las sesiones hasta llegar a ese ID de sesión. Una vez que llega a ese ID de sesión, el comando sale, independientemente de la cantidad de sesiones que se agregaron al servicio desde que se inició el comando. Por lo tanto, en el caso del Log Decoder de ejemplo, `sessions=200-now` inicia el procesamiento en la sesión 200, va hasta la sesión 1,000 y se cierra. Incluso si después de iniciarse el comando se agregan otros 1,000 registros al Log Decoder, aún se cierra después de procesar la sesión 1,000.
- El parámetro `sessions=now-u` significa comenzar en la última sesión y continuar con el procesamiento de todas las sesiones nuevas que entran. No procesa ninguna sesión existente (excepto la última), solo sesiones nuevas.

Para ver ejemplos de los comandos y lo que hacen, escriba `man sdk content examples` o consulte Ejemplos de comando SDK content.

Ejemplos de comando SDK content

El primer ejemplo de comando sdk content de NwConsole que aparece a continuación es simple y muestra todos los comandos que debe ingresar. Después de eso, los ejemplos solo muestran los comandos `sdk content`. El primer ejemplo crea un archivo de registro y explora los primeros 1,000 registros de una agregación de Concentrator desde un Log Decoder:

```
sdk open nw://admin:netwitness@myconcentrator.local:50005
sdk output /tmp
sdk content sessions=1-1000 render=logs append=mylogs.log
fileExt=.log
```

Este script genera 1,000 registros (suponiendo que en el servicio existe entre 1 y 1,000 sesiones) en el archivo **/tmp/mylogs.log**. Los registros están en texto crudo. El parámetro `fileExt=.log` es necesario para indicar al comando que se desea generar un archivo de registro.

```
sdk content sessions=1-1000 render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator=","
```

Este comando explora los mismos 1,000 registros como se indicó anteriormente, pero analiza el encabezado del registro y extrae la fecha y hora, el reenviador y otra información del registro, y los coloca en un archivo con formato CSV.

Ejemplo de CSV: 1422401778,10.250.142.64,10.25.50.66,hop04b-LC1,%MSISA-4: 81.136.243.248...

El registro de fecha y hora se expresa en tiempo Epoch. Los parámetros `includeHeaders` y `separator` solo se pueden usar en instalaciones de NetWitness Suite 10.4.0.2 y superior.

```
sdk content sessions=1-now render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator=","
where="risk.info='nw35120'"
```

Este comando escribe un archivo de registro en toda la gama de la sesión actual, pero solo registros que coinciden con `risk.info='nw35120'`. Tenga en cuenta que cuando agrega una cláusula Where, realiza una consulta en segundo plano para recopilar los ID de sesión para exportación. La consulta se debe ejecutar en un servicio con los campos adecuados indexados (generalmente es un Broker o un Concentrator). En este caso, puesto que está consultando el campo `risk.info`, vuelva a comprobar el servicio donde se ejecuta el comando para asegurarse de que esté indexado en el nivel de valor (IndexValues; consulte index-concentrator.xml para obtener ejemplos). De forma predeterminada, la mayoría de los Decoders solo tienen tiempo indexado. Si utiliza cualquier campo, excepto tiempo, en la cláusula Where, necesita transferir la consulta del Decoder a Concentrator, Broker o Archiver con los niveles de índice apropiados para la consulta. Puede encontrar más información sobre la indexación y la escritura de consultas en la *Guía de ajuste de la base de datos de NetWitness SuiteCore*.

```
sdk content sessions=1-now render=logs append=mylogs.log
fileExt=.log includeHeaders=true separator=","
where="threat.category exists && time='2015-01-05 15:00:00'-
'2015-01-05 16:00:00'"
```

Este comando es el mismo que el anterior, pero solo busca coincidencias de registros entre las 15:00 y las 16:00 h (UTC) el 5 de enero de 2015, que tengan una clave de metadatos `threat.category`. Nuevamente, debido a que esta consulta tiene un campo distinto de tiempo en la cláusula Where (`threat.category`), se debe ejecutar en un servicio con `threat.category` indexado por lo menos en el nivel `IndexKeys` (los operadores `exists` y `!exists` solo requieren un índice en el nivel de clave, a pesar de que los valores también funcionan bien).

```
sdk content sessions=1-now render=logs append=mylogs
fileExt=.log where="event.source begins 'microsoft'"
maxFileSize=1gb
```

Este comando crea varios archivos de registro, donde cada uno de ellos no tiene un tamaño mayor que 1 GiB. Antepone **mylogs** a los nombres de archivo y les agrega el registro de fecha y hora del primer paquete/registro en el archivo. Algunos nombres de archivo de ejemplo: **mylogs-1-2015-Jan-28T11_08_14.log**, **mylogs-2-2015-Jan-28T11_40_08.log** y **mylogs-3-2015-Jan-28T12_05_47.log**. En versiones anteriores a Security Analytics 10.5, el separador T entre la fecha y hora es un espacio.

```
sdk content sessions=1-now render=pcap append=mypackets
where="service=80,21 && time='2015-01-28 10:00:00'-'2015-01-28
15:00:00'" splitMinutes=5 fileExt=.pcap
```

Este comando explora todos los paquetes entre el período de cinco horas para tipos de servicios 80 y 21 y escribe un archivo PCAP. Cada 5 minutos, inicia un nuevo archivo PCAP.

```
sdk content time1="2015-01-28 14:00:00" time2="2015-01-28
14:15:00" render=pcap append=mydecoder fileExt=.pcap
maxFileSize=512mb sessions=1-now
```

Preste atención a este comando. ¿Por qué? Funciona en paquetes y registros, y es *extremadamente rápido*. La desventaja es que obtendrá todo entre los dos rangos de tiempo y no podrá usar una cláusula Where. Nuevamente, comienza a transmitir de regreso todo casi de inmediato y no requiere una consulta para ejecutar primero en el back-end. Debido a que todo se puede leer con I/O secuenciales, puede saturar completamente el vínculo de red entre el servidor y el cliente. Comienza a crear archivos que tienen antepuesto **mydecoder** y los divide en un archivo nuevo una vez que alcanzan un tamaño de 512 MIB.

```
sdk tailLogs
```

o (el comando equivalente):

```
sdk content render=pcap console=true sessions=now-u
```

Este es un comando divertido. Utiliza `sdk content` en segundo plano. El propósito de este comando es ver todos los registros entrantes en un Log Decoder. Eso es todo. Es muy simple. A medida que los registros entran en el Log Decoder (también puede ejecutarlo en un Broker o Concentrator), salen en la pantalla de la consola. Esta es una excelente manera de ver si el Log Decoder está capturando y qué es lo que exactamente ingresa en el Log Decoder. Este comando se ejecuta en modo continuo. No lo utilice si el Log Decoder está capturando en una alta tasa de recopilación (este comando no puede mantener el ritmo de él). Sin embargo, es útil para la verificación o con fines de solución de problemas.

```
sdk tailLogs where="device.id='ciscoasa'"
pathname=/mydir/anotherdir/mylogs
```

Este comando es el mismo que el anterior, excepto que solo genera registros que coinciden con la cláusula Where y en lugar de generar resultados en la consola, los escribe en un conjunto de archivos de registro en `/mydir/anotherdir` que no aumentan más de 1 GiB. Obviamente, también puede lograrlo con el comando `sdk content`, pero si prefiere el comportamiento predeterminado, debe escribir un poco menos.

```
sdk content sessions=now-u render=pcap where="service=80"
append=web-traffic fileExt=.pcap maxFileSize=2gb
maxDirSize=100gb
```

Este comando comienza a escribir PCAP de todo el tráfico web desde la sesión más reciente y todas las nuevas sesiones entrantes que coinciden con el servicio = 80. Escribe PCAP no mayores que 2 GiB y si todos los PCAP del directorio alcanzan un tamaño mayor que 100 GIB, elimina los PCAP más antiguos hasta que el directorio es 10 % menor que el tamaño máximo. Tenga en cuenta que la comprobación de tamaño del directorio no es exacta y solo comprueba cada 15 minutos de forma predeterminada. Puede ajustar la cantidad de minutos entre comprobaciones si pasa `cacheMinutes` como un parámetro, pero esto solo funciona con Security Analytics 10.5 y superior.

```
sdk content sessions=79000-79999 render=nwd
append=content-%1%.nwd metaFormatFilename=did
```

Este es el comando de respaldo de una persona inexperta. Explora 1,000 sesiones y genera el contenido completo (sesiones, metadatos, paquetes o registros) en formato NWD (formato de datos de NetWitness). NWD es un formato especial que se puede volver a importar a un paquete o Log Decoder sin volver a analizar. Por lo tanto, básicamente, la sesión analizada original se importa sin cambios. El registro de fecha y hora tampoco cambia, por lo que si originalmente se analizó hace 6 meses, durante la importación dicho registro se conservará como hace 6 meses.

Nota: No se debe esperar un gran rendimiento con este comando, especialmente con los paquetes. Recopilar los paquetes para una sesión implica una gran cantidad de I/O aleatorios, lo que puede ralentizar significativamente la exportación. Los registros no se ven muy afectados por este problema (un solo registro por sesión), pero en segundo plano este comando utiliza la API de /sdk content, la cual no es una API de transmisión orientada al rendimiento como los paquetes de /sdk. Por lo tanto, nuevamente, no debe esperar un gran rendimiento.

El parámetro `metaFormatFilename` es muy útil en este comando. Si este comando se ejecuta en un Concentrator con más de un servicio, se crearán los nombres de archivo NWD con los metadatos `did` para cada sesión (el %1% en el parámetro de anexo se sustituye por el valor de `did`). Cada nombre de archivo indicará exactamente de cuál Decoder provienen los datos.

```
sdk content session=1-u where="service=80,139,25,110"
render=files maxDirSize=200mb cacheMinutes=10
```

Este es otro comando divertido. Funciona muy similar a nuestro antiguo producto Visualize si vincula el directorio de salida con algo como el Explorador de Windows en modo de ícono. Extrae archivos de todo el tráfico de la web, correo electrónico y SMB. Esto incluye todos los tipos de archivos, como imágenes, archivos zip, videos, archivos PDF, documentos de Office, archivos de texto, archivos ejecutables y archivos de audio. Si extrae malware, el antivirus lo señalará. No se preocupe, el comando no ejecutará nada para que no infecte la máquina (a menos que intente ejecutarlo usted mismo). Sin embargo, puede ser útil porque si encuentra malware, el nombre de archivo indica el ID de la sesión de donde se extrajo. A continuación, puede consultar ese ID de sesión y ver qué host está posiblemente infectado por el malware y tomar medidas. Puede filtrar lo que se extrae con los parámetros `includeFileTypes` o `excludeFileTypes` (consulte la ayuda del comando). Por ejemplo, si se agrega `excludeFileTypes=".exe;.dmg;.msi"`, se evita que se extraigan archivos ejecutables e instaladores. Este comando ejecuta, sin interrupción, la extracción de archivos desde todas las sesiones existentes y desde cualquier sesión nueva. Después de que el directorio se abarrota con más de 200 MIB de archivos, se inicia automáticamente la limpieza de los archivos cada 10 minutos.

Nota: Este comando solo tiene sentido para las sesiones de paquete, no para los registros.

```
sdk content session=1-now where="time='2015-01-27 12:00:00'-'2015-01-27 13:00:00'
&& (service=25,110,80)" subdirFileTypes="audio=.wav;.mp3;.aac;
video=.wmv;.flv;.mp4;.mpg;.swf; documents=.doc;.xls;.pdf;.txt;.htm;.html
images=.png;.gif;.jpg;.jpeg;.bmp;.tif;.tiff archive=.zip;.rar; other=*"
renameFileTypes=".download|.octet-stream|.program|.exe;.jpeg|.jpg" render=files
maxDirSize=500mb
```

Este comando extrae archivos de sesiones HTTP y de correo electrónico desde un período de una hora y agrupa los archivos extraídos en los directorios que especifica el parámetro `subdirFileTypes`. Por ejemplo, cualquier archivo de audio extraído con la extensión .wav, .mp3 o .aac se colocará en el audio del subdirectorio, el cual se creará en el directorio de salida especificado. Lo mismo sucede con el resto de los grupos especificados en ese parámetro. Algunos archivos también se renombrarán automáticamente según su extensión de archivo. Esto se maneja mediante `renameFileTypes`. Cualquier archivo con una extensión .download, .octet-stream o .program se renombrará a .exe. Los archivos con la extensión .jpeg se renombrarán como .jpg. Una vez que el directorio de nivel superior supera los 500 MiB, se limpian los archivos más antiguos. Este comando se detiene en la última sesión cuando se inicia el comando.

```
sdk search session=1-now where="service=80,25,110" search="keyword='party' sp ci"
```

Este comando busca en todos los paquetes y los registros (el parámetro `sp`) la palabra clave `party`. Si party se encuentra en algún lugar de los paquetes o los registros, se presenta el ID de sesión junto con el texto que encontró y el texto que rodea al contexto. La cláusula Where indica que solo busca tráfico de correo electrónico y web. El parámetro `ci` significa que es una búsqueda que no distingue mayúsculas de minúsculas. Puede reemplazar `regex` por `keyword` para que lleve a cabo una búsqueda de regex.

```
sdk search session=1-now search="keyword='checkpoint' sp ci" render=log
append=checkpoint-logs.log fileExt=.log
```

Este es un ejemplo de comando interesante. Busca la palabra clave `checkpoint` en todos los registros (o podría ser paquetes) y si la encuentra, extrae el registro a un archivo **checkpoint-logs.log**. Existe todo tipo de posibilidades con este comando. Básicamente, cuando se detecta un acierto, transfiere la sesión a la llamada de contenido. Por lo tanto, todos los parámetros que pasa a `sdk search` y que no se reconocen se pasan junto con la llamada de contenido. Esto permite todas las funcionalidades de la llamada `sdk content`, pero solo cuando se trabaja en las sesiones con aciertos de búsqueda de contenido. Un gran poder conlleva una gran responsabilidad

Comandos utilizados para la solución de problemas

NwConsole proporciona los siguientes comandos que son útiles para la solución de problemas de Security Analytics:

- **whatIsWrong**: Proporciona una instantánea de la configuración, las estadísticas y los registros de fallas y advertencias de un servicio correspondientes a un período pasado especificado.
- **dbcheck**: Realiza la comprobación de coherencia de archivos de base de datos.
- **topQuery**: Ayuda a detectar consultas cuya ejecución tarda demasiado.
- **netbytes**: Soluciona problemas de las conexiones de red en el host actual
- **netspeed**: Soluciona problemas de conexión entre la computadora host en la cual se ejecuta NwConsole y la computadora remota conectada a él mediante el comando `login`.

Las siguientes secciones, así como las páginas de información de temas (man) y de ayuda de NwConsole, ofrecen información adicional.

whatIsWrong

Cuando un servicio no funciona correctamente, por lo general el motivo se encuentra en los registros que ha escrito el servicio. Puede usar el comando de la consola `whatIsWrong` para obtener una instantánea de la configuración, las estadísticas y los registros de fallas y advertencias de un servicio (con registros de contexto adyacentes) correspondientes a un período pasado especificado, el cual se fija de forma predeterminada en los siete días anteriores. Puede guardar los resultados de la ejecución de whatIsWrong en un archivo de texto crudo especificado. La salida de este comando puede ser un punto de partida útil para ayudar a determinar cuál es el problema actual de un servicio.

Para usar el comando de la consola whatIsWrong, inicie sesión en el servicio para solucionar problemas mediante el comando `login` y ejecute el comando `whatIsWrong`.

Sugerencia: Use `help whatIsWrong` para ver todos los parámetros disponibles, incluido el número de días u horas para buscar eventos, el nombre de ruta para almacenar los resultados, si desea o no anexar o sobrescribir el archivo de resultados y el delimitador que se utilizará en los campos de registro. También puede limitar la cantidad de registros más recientes que se utilizan para encontrar el contexto y puede especificar cuántos registros de contexto por registro de advertencia/falla desea recuperar.

Cada vez que reciba una solicitud de registros para un servicio principal, deberá ejecutar primero el comando whatIsWrong y usar los resultados recopilados como punto de partida.

dbcheck

El comando `dbcheck` se usa para realizar la comprobación de coherencia de archivos de base de datos (sesión, metadatos, paquetes, registros, estadísticas, etc.). Esto podría ser necesario cuando no se puede iniciar un servicio debido a errores de coherencia de los archivos de base de datos. Normalmente un servicio podría recuperar automáticamente y corregir los problemas de coherencia en el arranque, pero hay ocasiones en que esto no ocurre. Cuando se inicia un servicio (como Decoder), por lo general no lee ni abre la mayoría de los archivos de base de datos con el fin de iniciar rápidamente. Se supone la mayoría de los archivos está en un estado coherente y solo realiza una comprobación rápida de los archivos escritos más recientemente. Si hay problemas, `dbcheck` puede ejecutar las comprobaciones de coherencia, pero SOLO si el servicio no está en ejecución.

Precaución: No intente ejecutar este comando mientras se ejecuta un servicio.

Por ejemplo, puede comprobar un único archivo:

```
dbcheck /var/netwitness/decoder/packetdb/packet-000000001.nwpdb
```

También puede utilizar comodines para comprobar varios archivos:

```
dbcheck /var/netwitness/decoder/metadb/meta-00000002*.nwmdb
```

topQuery

El comando `topQuery` puede ayudar a detectar consultas cuya ejecución tarda demasiado. Este comando analiza los registros de auditoría de un servicio y devuelve las n consultas principales de mayor duración en un período de tiempo especificado.

La manera más fácil de ejecutar este comando es iniciar sesión en el servicio (por lo general, un Broker o un Concentrator) y escribir `topQuery`. El comportamiento predeterminado es devolver las 100 consultas principales de mayor duración en los últimos siete días.

Escriba `help topQuery` para obtener la lista de parámetros. Estos son algunos ejemplos adicionales con explicaciones:

```
topQuery hours=12 top=10
```

Este comando muestra las 10 consultas principales en las últimas 12 horas.

```
topQuery time1="2015-03-01 00:00:00" time2="2015-03-14 00:00:00"
```

Este comando muestra las 100 consultas principales entre el 1 de marzo de 2015 y el 14 de marzo de 2015. Las horas son en UTC, no en la hora local.

```
topQuery input=/var/log/messages output=/tmp/top20.txt top=20 user=sauser1
```

En lugar de conectarse a un servicio, analiza los mensajes de auditoría de syslog de las 20 consultas principales en los últimos 7 días, pero solo de las consultas ejecutadas por el usuario sauser1. Escribe las 20 consultas principales en /tmp/top20.txt en lugar de la pantalla de la consola. El parámetro user es una regex, de modo que puede especificar varios nombres de usuario si escribe algo similar a `user="(sauser1|sauser2)"`.

netbytes

El comando `netbytes` es muy útil para la solución de problemas de conexiones de red en el host actual. Muestra estadísticas continuas de envío y recepción para todas las interfaces de red. Una vez ejecutado, debe presionar **Ctrl-C** para salir de este comando, el cual también sale de NwConsole.

netspeed

El comando `netspeed` se usa para solucionar problemas de conexión entre la computadora host en la cual se ejecuta NwConsole y la computadora remota conectada a él mediante el comando `login`. Debe proporcionar la cantidad de bytes para la transferencia y se calculará la velocidad de la conexión. El comando netspeed es muy útil para solucionar problemas de rendimiento de agregación que podrían estar relacionados con la red.

```
login somedecoder:50004 admin ...
```

```
netspeed transfer=4g
```

Para solucionar problemas de conexión entre un Concentrator y un Decoder, acceda al Concentrator mediante el protocolo SSH, ejecute NwConsole y, a continuación, inicie sesión en el Decoder y ejecute `netspeed`. La salida del comando le ofrece una indicación del rendimiento máximo de la red. Si es mucho menor que la interfaz estándar de 1 GB/s, podría indicar un problema de red.