NXLog

Search docs

NXLog Documentation

⚠ You are viewing the documentation of our legacy products. Go to the NXLog Platform Documentation.

Browser history logs

NXLog Enterprise Edition exclusive feature

This guide explains how to collect browser history logs using NXLog.

Most popular browsers keep a log of the browsing history in an SQLite database. Information in this database includes the URL that was accessed, the title of the page, the time when the page was visited, and the number of times it was accessed. This data can be collected and processed with NXLog using the im_odbc module.

Browser history database location and format

The browsing history database is located in the user's profile folder and the path depends on the browser and operating system. This guide includes details of the databases used by Google Chrome, Mozilla Firefox, and Microsoft Edge browsers.

Google Chrome history location and details

Chrome history is stored in an SQLite database, the filename is `History` and can be found in the following locations:

Microsoft Windows Vista, 7, 8, 10, 11

```
C:\Users\<username>\AppData\Local\Google\Chrome\User Data\Default
```

Linux/Unix

RavenWhite EX2017
Home Depot v. RavenWhite
IPR2024-01316

The tables containing the Chrome browsing history in the *History* database are named *urls* and *visits*. Data from these tables can be joined together to retrieve the URL, page title, and time it was accessed.

Column	Type	Description
`id`	`INTEGER`	Primary Key - a unique ID for the record
`url`	`LONGVARCHAR`	The URL that was accessed
`title`	`LONGVARCHAR`	The title of the website
`visit_count`	`INTEGER`	The number of times the URL was accessed
`typed_count`	`INTEGER`	The number of times the user got to this website by typing the URL in the address bar
`last_visit_time`	`INTEGER`	Timestamp in nanoseconds when the website was last visited

Table 1. Chrome history - urls table structure

Column	Type	Description
`id`	`INTEGER`	A unique ID for the record
`url`	`INTEGER`	An ID corresponding to a record in the *urls* table
`visit_time`	`INTEGER`	Timestamp in nanoseconds when the website was visited

Table 2. Chrome history - visits table structure

id	url	title	visit_time
`100`	`https://www.nxlog.co`	High Performance Log Collection Solutions	`13250071947070670`

Table 3. Example data from joined visits and urls tables

Mozilla Firefox history location and details

Mozilla Firefox history is stored in an SQLite database, the file name is `places.sql` and can be found in the following locations:

Microsoft Windows Vista, 7, 8, 10, 11

```
C:\Users\<username>\AppData\Roaming\Mozilla\Firefox\Profiles\<profile folder>
```

```
/Users/<username>/Library/Application Support/Firefox/Profiles/<profile folder>
```

The tables containing the Firefox browsing history in the *places* database are named *moz_places* and *moz_historyvisits*. Data from these tables can be joined together to retrieve the URL, page title, and time it was accessed.

Column	Type	Description
`id`	`INTEGER`	Primary Key - a unique ID for the record
`url`	`LONGVARCHAR`	The URL that was accessed
`title`	`LONGVARCHAR`	The title of the website
`visit_count`	`INTEGER`	The number of times the URL was accessed
`typed`	`INTEGER`	The number of times the user got to this website by typing the URL in the address bar
`last_visit_date`	`INTEGER`	Timestamp in microseconds when the website was last visited

Table 4. Firefox history - moz_places table structure

Column	Type	Description
`id`	`INTEGER`	Primary Key - a unique ID for the record
`place_id`	`INTEGER`	An ID corresponding to a record in the *moz_places* table
`visit_date`	`INTEGER`	Timestamp in microseconds when the website was visited

Table 5. Firefox history - moz_historyvisits table structure

id	url	title	visit_date
`100`	`https://www.nxlog.co`	High Performance Log Collection Solutions	`1604932243415000`

Table 6. Example data from joined moz_historyvisits and moz_places tables

Microsoft Edge (v79+) history location and details

Microsoft Edge history is stored in an SQLite database, the database file name is `History` and can be found in the following location:

visits. Data from these tables can be joined together to retrieve the URL, page title, and the time it was accessed.

Column	Type	Description
`id`	`INTEGER`	Primary Key - a unique ID for the record
`url`	`LONGVARCHAR`	The URL that was accessed
`title`	`LONGVARCHAR`	The title of the website
`visit_count`	`INTEGER`	The number of times the URL was accessed
`typed_count`	`INTEGER`	The number of times the user got to this website by typing the URL in the address bar
`last_visit_time`	`INTEGER`	Timestamp in nanoseconds when the website was last visited

Table 7. Microsoft Edge history - urls table structure

Column	Type	Description
`id`	`INTEGER`	Primary Key - a unique ID for the record
`url`	`INTEGER`	An ID corresponding to a record in the *urls* table
`visit_time`	`INTEGER`	Timestamp in nanoseconds when the website was visited

Table 8. Microsoft Edge history - visits table structure

id	url	title	visit_time
`100`	`https://www.nxlog.co`	High Performance Log Collection Solutions	`13250071947070670`

Table 9. Example data from joined visits and urls tables

Collecting browser history logs

Example 1. Collecting Google Chrome browsing history logs on Windows with NXLog

This example configuration uses the xm_exec module to periodically run a batch script that copies the *History* database to a new location. This is done to avoid cases when the database is locked by

NXLog

nxlog.conf

```
<Extension json>
    Module              xm_json
</Extension>

<Extension exec>
    Module              xm_exec
    <Schedule>
        Every           30 min
        <Exec>
            odbc->module_stop();
            sleep(5000000);
            exec("C:\scripts\copy_chrome_db.cmd");
            odbc->module_start();
        </Exec>
    </Schedule>
</Extension>

<Input odbc>
    Module              im_odbc
    PollInterval        1200
    ConnectionString    DRIVER=SQLite3 ODBC Driver; \
                        Database=C:\logs\History_Chrome;Version=3;
    SQL                 SELECT visits.id AS id, \
                        DATETIME(ROUND(visits.visit_time / 1000000-
11644473600), \
                        'unixepoch', 'localtime') AS EventTime, \
                        urls.url AS URL, \
                        urls.title AS Title \
                        FROM visits \
                        INNER JOIN urls ON visits.url = urls.id \
                        WHERE visits.id > ?
    Exec                $Hostname = hostname();
    Exec                to_json();
</Input>
```

NOTE

Google Chrome saves the `visit_time` as a timestamp in nanoseconds. In this example, the SQL `DATETIME` function is used to convert it to local time.

copy_chrome_db.cmd

NXLog

NOTE

The cmd file needs to be saved in the path specified in the `Exec` block of the schedule in the configuration file. The user running NXLog needs to have permission to execute the file.

NOTE

The same configuration may be used for Microsoft Edge by changing the location of the original browser history database.

Output sample

```
{
  "id": 100,
  "EventTime": "2022-10-26 14:12:18",
  "URL": "https://nxlog.co",
  "Title": "High Performance Log Collection Solutions",
  "Hostname": "PC1",
  "EventReceivedTime": "2022-10-26T14:48:10.360819+03:00",
  "SourceModuleName": "odbc",
  "SourceModuleType": "im_odbc"
}
```

Example 2. Collecting Mozilla Firefox browsing history logs on Linux Ubuntu with NXLog

This example configuration uses the xm_exec module to periodically copy the `places.sqlite` database to a new location. This is done to avoid cases when the database is locked by Mozilla Firefox. The copied database is then processed by the im_odbc module. Since the database does not contain fields that identify the user, the event record is enriched by using the hostname() function to add the name of the machine to each record.

nxlog.conf

```
define FDBPATH /home/<user>/.mozilla/firefox/<profile_folder>/places.sqlite
define FDBCOPY /var/log/browser/places.sqlite

<Extension json>
    Module              xm_json
</Extension>

<Extension exec>
    Module              xm_exec
    <Schedule>
        Every           30 min
```

```
            odbc->module_start();
        </Exec>
    </Schedule>
</Extension>

<Input odbc>
    Module              im_odbc
    PollInterval        1200
    ConnectionString    DRIVER=SQLite3;Database=%FDBCOPY%;
    SQL                 SELECT moz_historyvisits.id AS id, \
                        DATETIME(ROUND(moz_historyvisits.visit_date / 1000000),
\
                        'unixepoch', 'localtime') AS EventTime, \
                        moz_places.url AS URL, \
                        moz_places.title AS Title \
                        FROM moz_historyvisits \
                        INNER JOIN moz_places ON \
                        moz_historyvisits.place_id = moz_places.id \
                        WHERE moz_historyvisits.id > ?
    Exec                $Hostname = hostname();
    Exec                to_json();
</Input>
```

NOTE

Mozilla Firefox saves the `visit_date` as a timestamp in microseconds. In this example, the SQL `DATETIME` function is used to convert it to local time.

Output sample

```
{
  "id": 100,
  "EventTime": "2022-10-26 18:45:12",
  "URL": "https://nxlog.co",
  "Title": "High Performance Log Collection Solutions",
  "Hostname": "PC1",
  "EventReceivedTime": "2022-10-26T18:56:57.272942+03:00",
  "SourceModuleName": "odbc",
  "SourceModuleType": "im_odbc"
}
```

Disclaimer

The accurateness of the content was tested and proved to be working in our lab environment at the time of the last revision with the following software versions:

NXLog version 5.6.7727
Ubuntu version 20.04.4 LTS
Microsoft Windows 11
Google Chrome version 107.0.5304.63
Mozilla Firefox version 106.0.2
Microsoft Edge version 106.0.1370.52

Last revision: 26 October 2022

Did you like this article? Please leave review about it

logo

Subscribe to our newsletter to get the latest updates, news, and products releases.

Email address

© Copyright 2024 NXLog FZE.
Privacy Policy. General Terms of Use

Follow us

Product

NXLog Platform
Integration
Professional Services
Plans

Resources