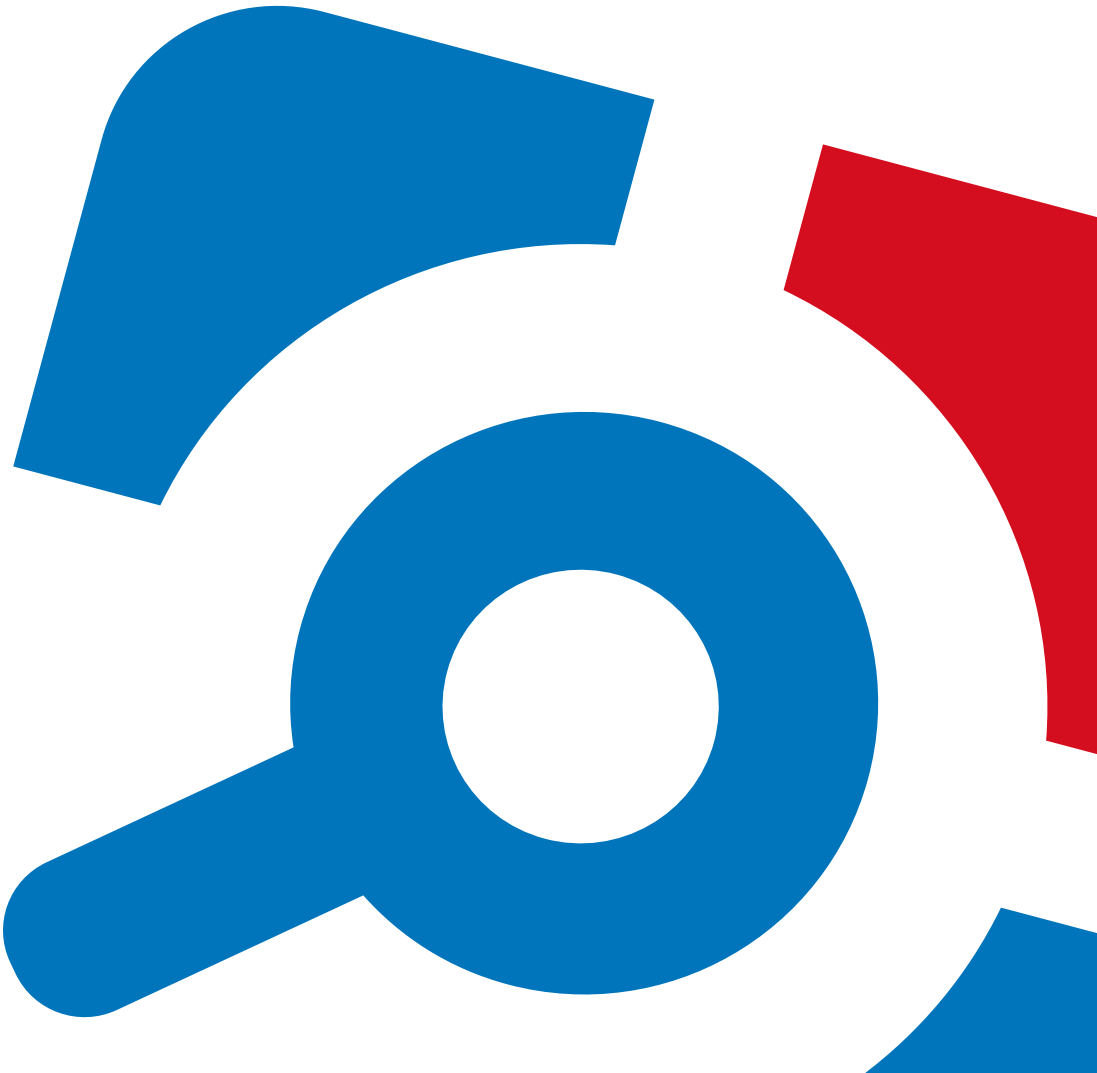
netwrix

Netwrix Auditor

Integration API Guide

Version: 8.0
6/3/2016



Legal Notice

The information in this publication is furnished for information use only, and does not constitute a commitment from Netwrix Corporation of any features or functions, as this publication may describe features or functionality not applicable to the product release or version you are using. Netwrix makes no representations or warranties about the Software beyond what is provided in the License Agreement. Netwrix Corporation assumes no responsibility or liability for the accuracy of the information presented, which is subject to change without notice. If you believe there is an error in this publication, please report it to us in writing.

Netwrix is a registered trademark of Netwrix Corporation. The Netwrix logo and all other Netwrix product or service names and slogans are registered trademarks or trademarks of Netwrix Corporation. Microsoft, Active Directory, Exchange, Exchange Online, Office 365, SharePoint, SQL Server, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks and registered trademarks are property of their respective owners.

Disclaimers

This document may contain information regarding the use and installation of non-Netwrix products. Please note that this information is provided as a courtesy to assist you. While Netwrix tries to ensure that this information accurately reflects the information provided by the supplier, please refer to the materials provided with any non-Netwrix product and contact the supplier for confirmation. Netwrix Corporation assumes no responsibility or liability for incorrect or incomplete information provided about non-Netwrix products.

Table of Contents

1. Introduction

This guide is intended for developers and provides instructions on how to use Netwrix Auditor Integration API. It suggests ideas for leveraging Netwrix Auditor audit data with third–party SIEM solutions, explains how to feed data from custom audit sources to the AuditArchive.

NOTE: Netwrix warns that Netwrix Auditor Integration API should be used by developers who have prior experience with RESTful architecture and solid understanding of HTTP protocol. Technology and tools overview is outside the scope of the current guide.

1.1. Netwrix Auditor Overview

Netwrix Auditor is an IT auditing platform that delivers complete visibility into changes and data access in hybrid cloud IT environments by providing actionable audit data about *who* changed *what*, *when* and *where* each change was made, and *who* has access to *what*. Netwrix Auditor helps organizations prevent security breaches caused by insider attacks, pass compliance audits with far less effort and expense, and keep tabs on what privileged users are doing in the environment.

Netwrix Auditor enables auditing of the broadest variety of IT systems, including Active Directory, Exchange, file servers, SharePoint, SQL Server, VMware and Windows Server. It also supports monitoring of privileged user activity in all other systems, even if they do not produce any logs, by enabling video recording of user screen activity and later search and replay. More than 160,000 IT departments worldwide rely on Netwrix Auditor to secure IT infrastructure, prove compliance and increase operational efficiency. The product has earned over 70 awards from leading industry publications, including SC Magazine, Windows IT Pro, Redmond Magazine and WindowSecurity.com.

Major benefits:

- **Change auditing and alerting:** Netwrix Auditor detects all configuration, content and security changes across your entire IT infrastructure. Reports and real-time alerts include the critical who, what, when and where details, including before and after values, enabling quick and effective response.
- **AuditIntelligence interactive search:** Netwrix Auditor enables you to easily search through audit data and fine-tune sorting and filtering criteria so you can quickly hone in on exactly the information you need.
- **Configuration assessment:** State-in-time™ reports show configuration settings at any point in time, such as group membership or password policy settings as they were configured a year ago.
- **Access auditing:** Monitoring of and reporting on successful and failed access to systems and data helps keep sensitive data safe.

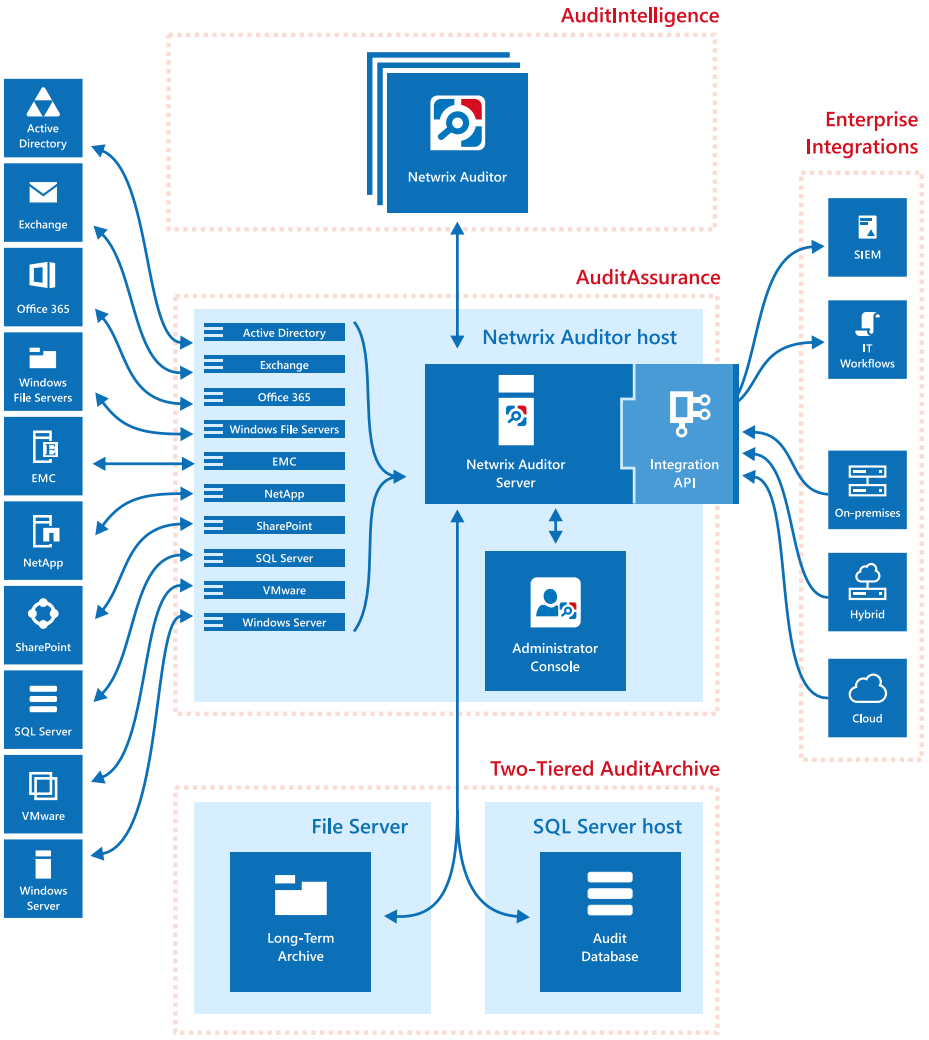
- **Predefined reports and diagrams:** Netwrix Auditor includes more than 150 predefined reports and diagrams. Reports can be exported to a range of formats, including PDF and XLS, and stakeholders can subscribe to reports to stay informed automatically by email.
- **AuditArchive™:** Netwrix Auditor's scalable two-tiered storage system (file-based + SQL database) holds consolidated audit data for more than 10 years.
- **Unified platform:** Many vendors require multiple standalone tools that are hard to integrate, but Netwrix Auditor is a unified platform that can audit the entire IT infrastructure.

Application	Features
Netwrix Auditor for Active Directory	Netwrix Auditor for Active Directory detects and reports on all changes made to the managed Active Directory domain, including AD objects, Group Policy configuration, directory partitions, and more. It makes daily snapshots of the managed domain structure that can be used to assess its state at present or at any moment in the past. The product provides logon activity summary, reports on interactive and non-interactive logons including failed logon attempts. Also, Netwrix Auditor for Active Directory helps detect and manage inactive users and expiring passwords. In addition, Netwrix Auditor for Active Directory provides a built–in Active Directory Object Restore tool that allows reverting unwanted changes to AD objects down to their attribute level.
Netwrix Auditor for Exchange	Netwrix Auditor for Exchange detects and reports on all changes made to Microsoft Exchange configuration and permissions. In addition, it tracks mailbox access events in the managed Exchange organization, and notifies the users whose mailboxes have been accessed by non–owners.
Netwrix Auditor for Office 365	Netwrix Auditor for Office 365 detects and reports on all changes made to Microsoft Exchange Online configuration and permissions. In addition, it tracks mailbox access events in the managed Exchange Online organization, and notifies the users whose mailboxes have been accessed by non–owners.
Netwrix Auditor for Windows File Servers	Netwrix Auditor for Windows File Servers detects and reports on all changes made to Windows–based file servers, including modifications of files, folders, shares and permissions, as well as failed and successful access attempts.
Netwrix Auditor for EMC	Netwrix Auditor for EMC detects and reports on all changes made to EMC Celerra, VNX/VNXe and Isilon storages, including modifications of files, folders, shares and permissions, as well as failed and successful

Application	Features
	access attempts.
Netwrix Auditor for NetApp	Netwrix Auditor for NetApp detects and reports on all changes made to NetApp Filer appliances both in cluster- and 7- modes, including modifications of files, folders, shares and permissions, as well as failed and successful access attempts.
Netwrix Auditor for SharePoint	Netwrix Auditor for SharePoint detects and reports on read access and changes made to SharePoint farms, servers and sites, including modifications of content, security settings and permissions.
Netwrix Auditor for SQL Server	Netwrix Auditor for SQL Server detects and reports on all changes to SQL Server configuration and database content.
Netwrix Auditor for VMware	Netwrix Auditor for VMware detects and reports on all changes made to ESX servers, folders, clusters, resource pools, virtual machines and their virtual hardware configuration.
Netwrix Auditor for Windows Server	Netwrix Auditor for Windows Server detects and reports on all changes made to Windows– based server configuration, including hardware devices, drivers, software, services, applications, networking settings, registry settings, DNS, and more. It also provides automatic consolidation and archiving of event logs data. Netwrix Auditor collects Windows event logs and syslog events from multiple computers across the network, stores them centrally in a compressed format, and enables convenient analysis of event log data. In addition, Netwrix Auditor for Windows Server can be configured to capture a video of users' activity on the audited computers.

1.2. How It Works

The image below provides overview of Netwrix Auditor architecture and gives a brief description of product components and incorporated technologies.



The **AuditIntelligence** technology is a brand new way of dealing with audit data, investigating incidents and enabling complete visibility across the entire IT infrastructure. **AuditIntelligence** is brought by the **Netwrix Auditor** client that provides easy access to audit data for IT managers, business analysts and other relevant employees via a straightforward and user-friendly interface. The **Netwrix Auditor** client allows generating reports, searching and browsing your audit data. You can install as many **Netwrix Auditor** clients as needed on workstations in your network, so that your authorized team members can benefit from using audit data collected by a single **Netwrix Auditor Server** to investigate issues and keep track of changes.

AuditAssurance is a technology that consolidates audit data from multiple independent sources (event logs, configuration snapshots, change history records, etc.). This allows detecting *who* changed *what*, *where* and *when* each change was made, and *who* has access to *what* even if one or several sources of information do not contain all of the required data, for example because it was deleted, overwritten, and so on.

AuditAssurance is provided by **Netwrix Auditor Server** and **Integration API**. **Netwrix Auditor Server** is a core part of **Netwrix Auditor** that collects, transfers and processes audit data. It contains several internal components responsible for gathering audit data from audited systems. **Netwrix Auditor Server** is managed with **Netwrix Auditor Administrator Console**, an interface for IT administrators designed to

configure IT infrastructure for auditing, define auditing scope, specify data collection, Audit Database and SMTP settings. **Netwrix Auditor Administrator Console** does not provide access to audit data. **Integration API** is a RESTful API that leverages audit data with custom on-premises or cloud data sources even if they are not supported as audited systems yet. API enables integration with third-party SIEM solutions by importing and exporting data to and from Netwrix Auditor.

Netwrix Auditor Server and **Integration API** interact with the **Two-Tiered AuditArchive** that is a scalable repository used for storing audit data collected by Netwrix Auditor and imported from other data sources and IT systems using **Integration API**. The **Two-Tiered AuditArchive** includes:

- The file-based **Long-Term Archive**
- The SQL-based short-term **Audit Database**

2. Netwrix Auditor Integration API Overview

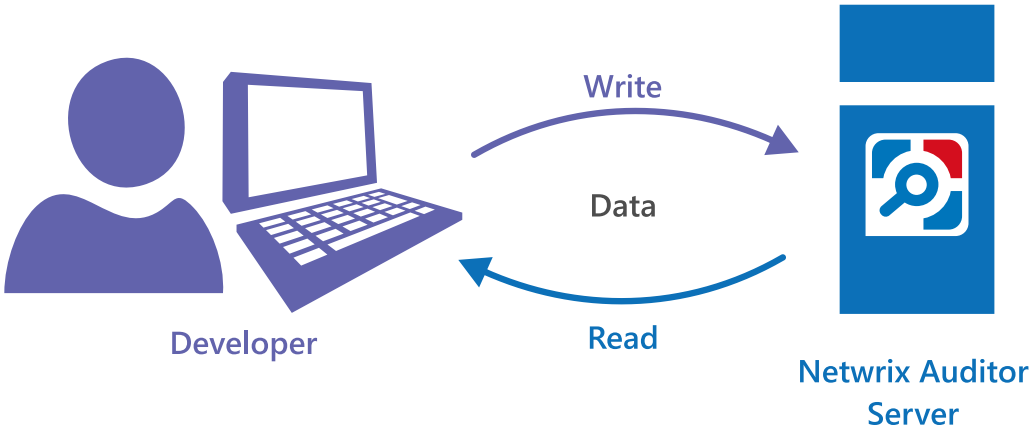
Netwrix Auditor Integration API—endless integration, auditing and reporting capabilities.

The Netwrix Auditor Integration API provides access to audit data collected by Netwrix Auditor through REST API endpoints. According to the RESTful model, each operation is associated with a URL. Integration API provides the following capabilities:

- **Data in**: Centralize auditing and reporting by feeding Netwrix Auditor with audit data from any existing on-premises or cloud applications. All of your audit data will be centrally stored and ready for reporting.
- **Data out**: Get the most from your SIEM investment by feeding more granular audit data into your HP Arcsight, Splunk, IBM QRadar or other solution, thus increasing the signal-to-noise ratio. Moreover, you can also feed the granular audit data from Netwrix Auditor into critical IT processes, such as change management or ticketing, to further automate and streamline operations.

Netwrix Auditor Integration API operates with XML- and JSON-formatted Activity Records—minimal chunks of audit data containing information on *who* changed *what*, *when* and *where* this change was made. XML format is set as default.

With Integration API you can write Activity Records to the SQL Server-based Audit Database located on Netwrix Auditor Server and access audit data from remote computers. Also, Netwrix prepares sample scripts to help you integrate your SIEM solutions with Netwrix Auditor.



Netwrix does not limit you with applications that can be used with Integration API. You can write RESTful requests using any tool or application you prefer —cURL, Telerik Fiddler, various Google Chrome or Mozilla FireFox plug-ins, etc.

3. Prerequisites

3.1. Configure Integration API Settings

The **Netwrix Auditor Integration API Service** responsible for processing API requests is installed along with Netwrix Auditor and is enabled. By default, for communication Netwrix Auditor Integration API uses HTTPS with an automatically generated certificate and port 9699.

Refer to Security for detailed instructions on how to disable HTTPS and manage other API settings.

To change port

1. In Netwrix Auditor Administrator Console, navigate to **Settings** → **Integration API**.
2. Make sure **Enable Integration API** is checked.
3. Click **Modify**. Windows firewall rule will be automatically created.

   NOTE: If you use a third-party firewall, you must create a rule for inbound connections manually.

3.2. Configure Audit Database Settings

When you first configure the Audit Database settings in Netwrix Auditor Administrator Console, the product creates configuration databases including **Netwrix_Auditor_API**. This database is designed to store data imported from the other sources using Netwrix Auditor Integration API.

Make sure the Audit Database settings are configured in Netwrix Auditor Administrator Console. To check or configure these settings, navigate to **AuditArchive** → **Audit Database**.

NOTE: You cannot use Netwrix Auditor Integration API without configuring the Audit Database.

See Netwrix Auditor Administrator's Guide for detailed instructions on how to configure SQL Server settings.

4. API Endpoints

Method	Endpoint	POST Data	Description
GET	/netwrix/api/v1/activity_records/enum	—	Returns Activity Records. See Retrieve Activity Records for more information.
POST	/netwrix/api/v1/activity_records/enum	Continuation Mark	Returns next 1,000 Activity Records. See Continuation Mark for more information.
POST	/netwrix/api/v1/activity_records/search	Search Parameters	Returns Activity Records matching a criteria defined in search parameters. See Search Activity Records for more information.
POST	/netwrix/api/v1/activity_records/	Activity Records	Writes data to the Audit Database. See Write Activity Records for more information.

5. Authentication

Authentication is required for all endpoints. The following authentication methods are supported:

- NTLM—recommended

  NOTE: If NTLM authentication is disabled through a group policy, you will not be able to address Netwrix Auditor Server by its IP address.

- Negotiate
- Digest
- Basic

To run requests to Netwrix Auditor Integration API, the account must be a member of the **Netwrix Auditor Client Users** group on the computer that hosts Netwrix Auditor Server.

Review the example below to see how to authenticate in cURL:

- `curl https://172.28.6.15:9699/netwrix/api/v1/activity_records/enum -u Enterprise\NetwrixUser:NetwrixIsCool`

6. Retrieve Activity Records

6.1. Endpoint

Use to export data from the Audit Database. By default, first 1,000 Activity Records are returned. To get the next Activity Records, send a POST request to the same endpoint containing a Continuation mark.

Method	Endpoint	POST Data
GET	`https://{host:port}/netwrix/api/v1/activity_records/enum{?format=json}{&count=Number}`	—
POST	`https://{host:port}/netwrix/api/v1/activity_records/enum{?format=json}{&count=Number}`	Continuation Mark

6.2. Request Parameters

Parameter	Mandatory	Description
`host:port`	Yes	Replace with the IP address or a name of your Netwrix Auditor Server host and port (e.g., *172.28.6.15:9699*, *stationwin12:9699*, *WKSWin2012.enterprise.local:9699*). **NOTE:** With enabled HTTPS, provide the computer name as it appears in certificate properties.
`format=json`	No	Add this parameter to retrieve data in JSON format. Otherwise, XML-formatted Activity Records will be returned.
`count=Number`	No	Add this parameter to define the number of Activity Records to be exported. Replace `Number` with a number (e.g., `&count=1500`).

NOTE: Optional parameters (format and count) can be provided in any order. The first parameter must start with **?**, others are joined with **&**, no spaces required (e.g., `?format=json&count=1500`).

6.3. Response

Request Status	Response
Success	The HTTP status code in the response header is **200 OK**. The response body

Request Status	Response

	contains Activity Records and Continuation Mark.
	`HTTP/1.1 200 OK`
	`Server: Microsoft-HTTPAPI/2.0`
	`Content-Length: 311896`
	`Content-Type: application/xml`
	`Date: Fri, 08 Apr 2016 13:56:22 GMT`
	or
	`HTTP/1.1 200 OK`
	`Server: Microsoft-HTTPAPI/2.0`
	`Content-Length: 311896`
	`Content-Type: application/json`
	`Date: Fri, 08 Apr 2016 13:56:22 GMT`

Error	The header status code is an error code. Depending on the error code, the response body may contain an error object. See Response Status Codes for more information.

6.4. Usage Example—Retrieve All Activity Records

This example describes how to retrieve all Activity Records from the Audit Database.

1. Send a GET request. For example:

Format	Request
XML	`curl https://WKSWin2012:9699/netwrix/api/v1/activity_records/enum -u Enterprise\NetwrixUser:NetwrixIsCool`
JSON	`curl https://WKSWin2012:9699/netwrix/api/v1/activity_records/enum?format=json -u Enterprise\NetwrixUser:NetwrixIsCool`

2. Receive the response. Below is an example of a successful GET request. The status is **200 OK**. For XML, a response body contains the `ActivityRecordList` root element with Activity Records and a Continuation mark inside. For JSON, a response body contains the `ActivityRecordList` array with Activity Records collected in braces {} and a Continuation mark.

XML

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9
  IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9Ij
  Q0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlB
  Q0YxNzJFQyIvPjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <AuditedSystem>Active Directory</AuditedSystem>
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
```

```
    <Action>Added</Action>
    <When>2016-02-14T15:42:34Z</When>
    <Where>EnterpriseDC1.enterprise.local</Where>
    <Who>ENTERPRISE\Administrator</Who>
    <Workstation>EnterpriseDC1.enterprise.local</Workstation>
  </ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
</ActivityRecordList>
```

JSON

```
{
  "ActivityRecordList": [
    {
      "Action": "Added",
      "AuditedSystem": "Active Directory",
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
      "When": "2016-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
  ],
  "ContinuationMark": "PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCO
  DF9IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY
  9IjQ0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5R
  jlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A"
}
```

3. Continue retrieving Activity Records. Send a POST request containing this Continuation mark to the same endpoint. See Continuation Mark for more information.

XML

```
curl -H "Content-Type: application/xml; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/enum -u
Enterprise\NetwrixUser:NetwrixIsCool --data-binary
@C:\APIdocs\ContMark.xml
```

```
<?xml version="1.0" standalone="yes"?>
<ContinuationMark xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9IiB0PSJDb250aW51YX
  Rpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUxMzU2MTE5MDcw
```

```
  NzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyIvPjwvYT
  48L24+PC9ucj4A
</ContinuationMark>
```

JSON

```
curl -H "Content-Type: application/json; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/enum?format=json
-u Enterprise\NetwrixUser:NetwrixIsCool --data-binary
@C:\APIdocs\ContMark.json
```

```
"PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCODF9IiB0PSJDb250aW51YX
Rpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUxMzU2MTE5MDcwN
zcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyIvPjwvYT48
L24+PC9ucj4A"
```

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

4. Receive the next response. On success, the status is **200 OK**. For XML, a response body contains the `ActivityRecordList` root element with next Activity Records and a new Continuation mark inside. For JSON, a response body contains the `ActivityRecordSearch` array with next Activity Records collected in braces {} and a new Continuation mark.
5. Continue retrieving Activity Records. Send POST requests containing new Continuation marks until you receive a **200 OK** response with no Activity Records inside the `ActivityRecordList`. It means you reached the end of the Audit Database.

7. Search Activity Records

The search functionality in the Netwrix Auditor Integration API reproduces AuditIntelligence Search in the Netwrix Auditor client. See Netwrix Auditor User Guide for detailed instruction on how to search and filter audit data.

As AuditIntelligence Search in the Netwrix Auditor client, this REST API endpoint allows you to retrieve Activity Records matching a certain criteria. You can create your own set of filters in the Search parameters file. See Search Parameters for more information.

7.1. Endpoint

To retrieve Activity Records matching a certain criteria, send a POST request containing search parameters (also may include a Continuation mark). See Search Parameters for more information.

Method	Endpoint	POST Data
`POST`	`https://{host:port}/netwrix/api/v1/activity_records/search{?format=json}{&count=Number}`	Search Parameters

7.2. Request Parameters

Parameter	Mandatory	Description
`host:port`	Yes	Replace with the IP address or a name of your Netwrix Auditor Server host and port (e.g., *172.28.6.15:9699*, *stationwin12:9699*, *WKSWin2012.enterprise.local:9699*). **NOTE:** With enabled HTTPS, provide the computer name as it appears in certificate properties.
`format=json`	No	Add this parameter to retrieve data in JSON format. Otherwise, XML-formatted Activity Records will be returned.
`count=Number`	No	Add this parameter to define the number of Activity Records to be exported. Replace `Number` with a number (e.g., `?count=1500`).

NOTE: Optional parameters (format and count) can be provided in any order. The first parameter must start with **?**, others are joined with **&**, no spaces required (e.g., `?format=json&count=1500`).

7.3. Response

Request Status	Response

Success	The HTTP status code in the response header is **200 OK**. The response body contains Activity Records and Continuation Mark.
	`HTTP/1.1 200 OK`
	`Server: Microsoft-HTTPAPI/2.0`
	`Content-Length: 311896`
	`Content-Type: application/xml`
	`Date: Fri, 08 Apr 2016 13:56:22 GMT`
	or
	`HTTP/1.1 200 OK`
	`Server: Microsoft-HTTPAPI/2.0`
	`Content-Length: 311896`
	`Content-Type: application/json`
	`Date: Fri, 08 Apr 2016 13:56:22 GMT`

Error	The header status code is an error code. Depending on the error code, the response body may contain an error object. See Response Status Codes for more information.

7.4. Usage Example—Retrieve All Activity Records Matching Search Criteria

This example describes how to retrieve all Activity Records matching search criteria.

1. Send a POST request containing search parameters. See Search Parameters for more information.

   For example, this request retrieves Activity Records where administrator added new objects to the Active Directory domain. Groups and group policies are not taken into account. Changes could only occur between September 16, 2015 and March 16, 2016.

   XML

```
curl -H "Content-Type:application/xml; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/search -u
Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Search.xml
```

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
 <FilterList>
  <Who>Administrator</Who>
  <AuditedSystem>Active Directory</AuditedSystem>
  <Action>Added</Action>
  <ObjectType Operator="DoesNotContain">Group</ObjectType>
  <When>
   <From>2015-09-16T16:30:00+11:00</From>
   <To>2016-03-16T00:00:00Z</To>
  </When>
```

```
  </FilterList>
</ActivityRecordSearch>
```

JSON

```
curl -H "Content-Type:application/json; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/
search?format=json -u Enterprise\NetwrixUser:NetwrixIsCool --data-binary
@C:\APIdocs\Search.json
```

```
{
  "FilterList": {
    "Who": "Administrator",
    "AuditedSystem": "Active Directory",
    "Action": "Added",
    "ObjectType": { "DoesNotContain": "Group"},
    "When": {
      "From": "2015-09-16T16:30:00+11:00",
      "To": "2016-03-16T00:00:00Z"
    }
  }
}
```

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

2. Receive the response. Below is an example of a successful search request. The status is **200 OK**. For XML, a response body contains the `ActivityRecordList` root element with Activity Records matching filter criteria and a Continuation mark inside. For JSON, a response body contains the `ActivityRecordList` array with Activity Records matching filter criteria and collected in braces {}, and a Continuation mark.

XML

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9
  IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9Ij
  Q0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlB
  Q0YxNzJFQyIvPjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <AuditedSystem>Active Directory</AuditedSystem>
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
    <Action>Added</Action>
    <When>2016-02-14T15:42:34Z</When>
```

```
    <Where>EnterpriseDC1.enterprise.local</Where>
    <Who>ENTERPRISE\Administrator</Who>
    <Workstation>EnterpriseDC1.enterprise.local</Workstation>
  </ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
</ActivityRecordList>
```

JSON

```
{
  "ActivityRecordList": [
    {
      "Action": "Added",
      "AuditedSystem": "Active Directory",
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
      "When": "2016-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
  ],
  "ContinuationMark": "PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCO
  DF9IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY
  9IjQ0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5R
  jlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A"
}
```

3. Continue retrieving Activity Records. Send a POST request containing your search parameters and this Continuation mark to the same endpoint. See Continuation Mark for more information.

XML

```
curl -H "Content-Type:application/xml; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/search -u
Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Search.xml
```

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9
  IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9Ij
  Q0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlB
  Q0YxNzJFQyIvPjwvYT48L24+PC9ucj4A</ContinuationMark>
  <FilterList>
```

```
    <Who>Administrator</Who>
    <AuditedSystem>Active Directory</AuditedSystem>
    <Action>Added</Action>
    <ObjectType Operator="DoesNotContain">Group</ObjectType>
    <When>
      <From>2015-09-16T16:30:00+11:00</From>
      <To>2016-03-16T00:00:00Z</To>
    </When>
  </FilterList>
</ActivityRecordSearch>
```

JSON

```
curl -H "Content-Type:application/json; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_
records/search?format=json -u Enterprise\NetwrixUser:NetwrixIsCool --
data-binary @C:\APIdocs\Search.json
```

```
{
  "ContinuationMark": "PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCO
  DF9IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY
  9IjQ0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5R
  jlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A",
  "FilterList": {
    "Who": "Administrator",
    "AuditedSystem": "Active Directory",
    "Action": "Added",
    "ObjectType": { "DoesNotContain": "Group"},
    "When": {
      "From": "2015-09-16T16:30:00+11:00",
      "To": "2016-03-16T00:00:00Z"
    }
  }
}
```

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

4. Receive the next response. On success, the status is **200 OK**. For XML, a response body contains the `ActivityRecordList` root element with next Activity Records and a new Continuation mark inside. For JSON, a response body contains the `ActivityRecordSearch` array with next Activity Records collected in braces {} and a new Continuation mark.

5. Continue retrieving Activity Records. Send POST requests containing your search parameters with new Continuation marks until you receive a **200 OK** response with no Activity Records inside the `ActivityRecordList`. It means you retrieved all Activity Records matching your search criteria.

8. Write Activity Records

8.1. Endpoint

Write data to the **Netwrix_Auditor_API** database located in the Audit Database and to the Long-Term Archive. To feed data send a POST request containing Activity Records.

After feeding data to the Audit Database it will become searchable in the Netwrix Auditor client and through /netwrix/api/v1/activity_records/search and /netwrix/api/v1/activity_records/enum endpoints.

Method	Endpoint	POST Data
`POST`	`https://{host:port}/netwrix/api/v1/activity_records/{?format=json}`	Activity Records

NOTE: Netwrix recommends limiting the input Activity Records file to 50MB and maximum 1,000 Activity Records.

8.2. Request Parameters

Parameter	Mandatory	Description
`host:port`	Yes	Replace with the IP address or a name of your Netwrix Auditor Server host and port (e.g., *172.28.6.15:9699*, *stationwin12:9699*, *WKSWin2012.enterprise.local:9699*). **NOTE:** With enabled HTTPS, provide the computer name as it appears in certificate properties.
`?format=json`	No	Add this parameter to write data in JSON format. Otherwise, Netwrix Auditor Server will expect XML-formatted Activity Records and will consider JSON invalid.

8.3. Response

Request Status	Response
Success	The HTTP status code in the response header is **200 OK** and the body is empty. `HTTP/1.1 200 OK`

Request Status	Response
	Server: Microsoft-HTTPAPI/2.0 Content-Length: 0 Content-Type: text/plain Date: Fri, 08 Apr 2016 13:56:22 GMT
Error	The header status code is an error code. Depending on the error code, the response body may contain an error object. See Response Status Codes for more information.

8.4. Usage Example—Write Data

This example describes how to feed Activity Records to the Audit Database.

1. Send a POST request containing Activity Records. See Activity Records for more information. For example:

XML

```
curl -H "Content-Type:application/xml; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/ -u
Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Input.xml
```

```
<?xml version="1.0" encoding="utf-8"?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ActivityRecord>
    <Who>Admin</Who>
    <ObjectType>Stored Procedure</ObjectType>
    <Action>Added</Action>
    <What>Databases\ReportServer\Stored Procedures\dbo.sp_New</What>
    <Where>WKSWin12SQL</Where>
    <When>2016-02-19T03:43:49-11:00</When>
  </ActivityRecord>
  <ActivityRecord>
    <Action>Modified</Action>
    <ObjectType>Mailbox</ObjectType>
    <What>Shared Mailbox</What>
    <When>2016-02-10T14:46:00Z</When>
    <Where>BLUPR05MB1940</Where>
    <Who>admin@enterprise.onmicrosoft.com</Who>
    <DetailList>
      <Detail>
        <PropertyName>Custom_Attribute</PropertyName>
        <Before>1</Before>
        <After>2</After>
```

```
    </Detail>
   </DetailList>
  </ActivityRecord>
</ActivityRecordList>
```

JSON

```
curl -H "Content-Type:application/json; Charset=UTF-8"
https://WKSWin2012:9699/netwrix/api/v1/activity_records/?format=json -u
Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Input.json
```

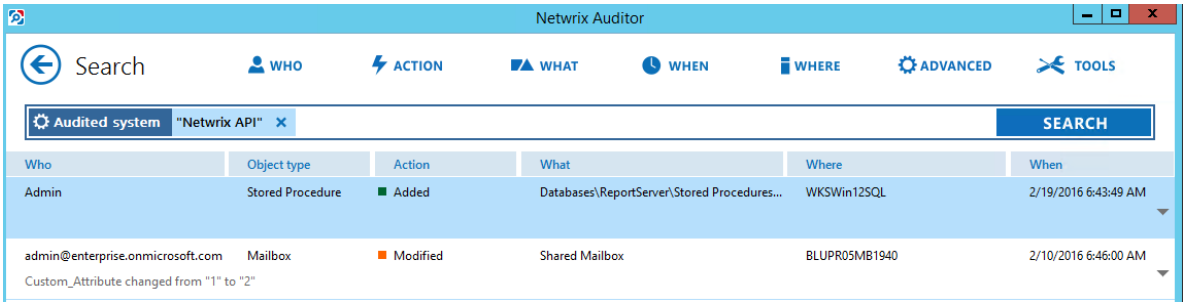
```
[
  {
    "Who": "Admin",
    "ObjectType": "Stored Procedure",
    "Action": "Added",
    "What": "Databases\\ReportServer\\Stored Procedures\\dbo.sp_New",
    "Where": "WKSWin12SQL",
    "When": "2016-02-19T03:43:49-11:00"
  },
  {
    "Action": "Modified",
    "ObjectType": "Mailbox",
    "What": "Shared Mailbox",
    "When": "2016-02-10T14:46:00Z",
    "Where": "BLUPR05MB1940",
    "Who": "admin@enterprise.onmicrosoft.com",
    "DetailList": [
      {
        "PropertyName": "Custom_Attribute",
        "Before": "1",
        "After": "2"
      }
    ]
  }
]
```

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

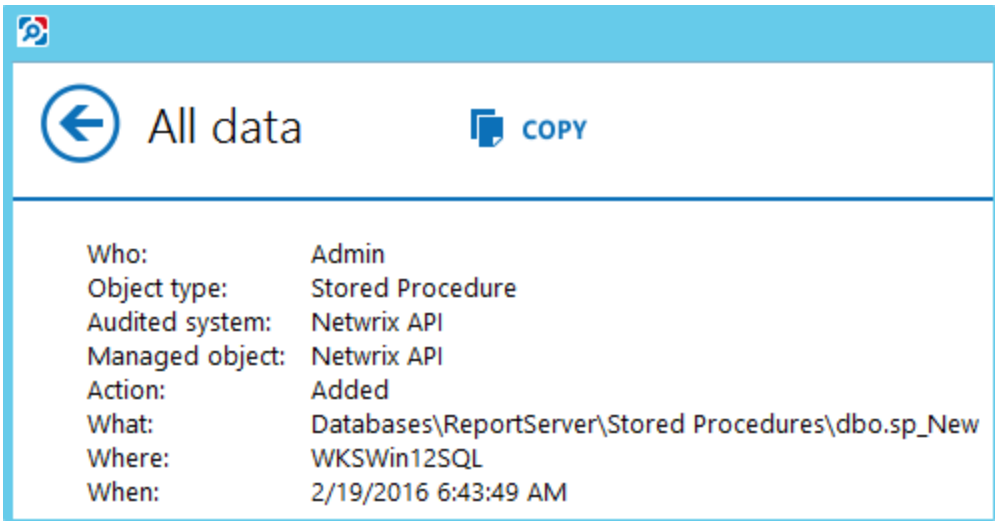
2. Receive the response. Below is an example of a successful write request. The status is **200 OK** and the body is empty.

```
HTTP/1.1 200 OK
Server: Microsoft-HTTPAPI/2.0
Content-Length: 0
Content-Type: text/plain
Date: Fri, 08 Apr 2016 13:56:22 GMT
```

3. Send more POST requests containing Activity Records if necessary.
4. Check that posted data is now available in the Audit Database. Run a search request to /netwrix/api/v1/activity_records/search endpoint or use AuditIntelligence Search in the Netwrix Auditor client. For example:



NOTE: For input Activity Records, the Audited system and Managed object are automatically set to **Netwrix API**.



9. Post Data

While running requests to Netwrix Auditor Integration API endpoints, you will need to post data, e.g., a Continuation mark in order to continue retrieving Activity Records, Search parameters to find Activity Records matching your search, or Activity Records you want to feed to the Audit Database. Data is sent in the request body and must be formatted according to XML convention and compatible with Netwrix-provided XSD schemas.

NOTE: The file must be formatted in accordance with XML standard. The following symbols must be replaced with corresponding XML entities: **&** (ampersand), **<** (less than), and **>** (greater than) symbols.

Symbol	XML entity
&	**&**
e.g., Ally **&** Sons	e.g., Ally **&** Sons
<	**<**
e.g., CompanyDC<100	e.g., CompanyDC**<**100
>	**>**
e.g., ID>500	e.g., ID**>**500

Also, Netwrix allows transferring data in JSON format (organized as name and value pairs). JSON file must be formatted in accordance with JSON specification. Special characters in JSON strings must be preceded with the **** character: **"** (double quotes), **/** (slash), **** (backslash). E.g., "\\local\\enterprise\\Users\\Jason Smith"". Trailing comma is not supported.

Review the following for additional information:

- Continuation Mark
- Search Parameters
- Activity Records

9.1. Continuation Mark

When exporting data from the Audit Database, a successful response includes:

- For XML—A `<ContinuationMark>` inside the `<ActivityRecordsList>` root element.
- For JSON—An object with the "ContinuationMark" field.

Continuation mark is a checkpoint, use it to retrieve data starting with the next Activity Record.

Send a POST request containing Continuation mark to the following endpoints:

Method	Endpoint	Description
POST	/netwrix/api/v1/activity_records/enum	Returns next Activity Records.
POST	/netwrix/api/v1/activity_records/search	Returns next Activity Records matching a filter criteria.

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

You can send as many POST requests as you want. A new response returns next Activity Records and a new Continuation mark. Once all the Activity Records are retrieved, you will receive a **200 OK** response with no Activity Records inside the `ActivityRecordList` root element (XML) or array (JSON).

9.1.1. Schema

Copy the contents of `ContinuationMark` to a separate XML or JSON file (e.g., ContMark.xml).

Format	Schema description

XML	The file must be compatible with the XML schema. On computer where Netwrix Auditor Administrator Console is installed, you can find XSD file under *Netwrix_Auditor_installation_folder\Audit Core\API Schemas*.
	`<?xml version="1.0" encoding="utf-8"?>`
	`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://schemas.netwrix.com/api/v1/activity_records/" xmlns="http://schemas.netwrix.com/api/v1/activity_records/" elementFormDefault="qualified">`
	`<xs:element name="ContinuationMark" type="xs:string"/>`
	`</xs:schema>`
	The `ContinuationMark` root element contains a value previously returned by Netwrix Auditor Integration API.

JSON	JSON-formatted Continuation mark includes the field value in quotes.

If you want to retrieve next Activity Records for your search, include the Continuation mark to your Search parameters file. See Search Parameters for more information.

9.1.2. Example

XML

Retrieve Activity Records

```
<?xml version="1.0" standalone="yes"?>
<ContinuationMark xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9IiB0PSJDb250aW51YXRpb2
  5NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUxMzU2MTE5MDcwNzcyMjE6M
  jAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A
</ContinuationMark>
```

Search Activity Records

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9IiB0
  PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUxMzU
  2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyIvPj
  wvYT48L24+PC9ucj4A</ContinuationMark>
  <FilterList>
    <Who>Administrator</Who>
    <AuditedSystem>Active Directory</AuditedSystem>
    <Action>Added</Action>
    <ObjectType Operator="DoesNotContain">Group</ObjectType>
    <When>
      <From>2015-09-16T16:30:00+11:00</From>
      <To>2016-03-16T00:00:00Z</To>
    </When>
  </FilterList>
</ActivityRecordSearch>
```

JSON

Retrieve Activity Records

```
"PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCODF9 IiB0PSJDb250aW51YXRpb2
5NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUxMzU2TE5MDcwNzcyMjE6MjA
xNjA0MDUxMDM3NDkxMjVGMUJ FRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A"
```

Search Activity Records

```
{
  "ContinuationMark": "PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCODF9I
  iB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9IjQ0MzUx
  MzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlBQ0YxNzJFQyI
  vPjwvYT48L24+PC9ucj4A",
  "FilterList": {
    "Who": "Administrator",
    "AuditedSystem": "Active Directory",
```

```
    "Action": "Added",
    "ObjectType": { "DoesNotContain": "Group"},
    "When": {
      "From": "2015-09-16T16:30:00+11:00",
      "To": "2016-03-16T00:00:00Z"
    }
  }
}
```

9.2. Search Parameters

Send the search parameters in the POST request body to narrow down the search results returned by the /netwrix/api/v1/activity_records/search endpoint. The Search parameters file includes one or more filters with operators and values (e.g., to find entries where *Audited System* is *SharePoint*); it may also contain a Continuation Mark. Generally, the Search parameters file looks similar to the following:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<ActivityRecordSearch xmlns="http//schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>Continuation mark</ContinuationMark>
  <FilterList>
    <Filter1>Value</Filter1>
    <Filter2>Value1</Filter2>
    <Filter2>Value2</Filter2>
    <Filter3 Operator="MatchType1">Value1</Filter3>
    <Filter3 Operator="MatchType2">Value2</Filter3>
    <Filter4>Value1</Filter4>
    <Filter4 Operator="MacthType">Value2</Filter4>
  </FilterList>
</ActivityRecordSearch>
```

JSON

```
{
  "ContinuationMark": "Continuation Mark",
  "FilterList": {
    "Filter1": "Value",
    "Filter2": [ "Value1", "Value2" ],
    "Filter3": {
    "MatchType1": "Value1",
    "MatchType2": "Value2"
    },
    "Filter4": [ "Value1", { "MatchType": "Value2" } ]
  }
}
```

NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

9.2.1. Schema

Format	Schema description
XML	The file must be compatible with the XML schema. On computer where Netwrix Auditor Administrator Console is installed, you can find XSD file under *Netwrix_Auditor_installation_folder\Audit Core\API Schemas*. The `ActivityRecordSearch` root element includes the `FilterList` element with one or more `Filter` elements inside. The root element may contain a `ContinuationMark` element. Each `Filter` specified within the `FilterList` must have a value to search for. The element may also include a modifier—a match type operator. **NOTE:** **minOccurs="0"** indicates that element is optional and may be absent in the Search parameters.
JSON	The `FilterList` object includes with one or more `Filter` entries inside. JSON may contain a `ContinuationMark` object. Each `Filter` specified within the `FilterList` must have a value to search for. The entry may also include a modifier—a match type operator.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://schemas.netwrix.com/api/v1/activity_records/"
           xmlns="http://schemas.netwrix.com/api/v1/activity_records/"
           elementFormDefault="qualified">

  <xs:complexType name="StringFilter">...</xs:complexType>

  <xs:simpleType name="ActionEnum">...</xs:simpleType>

  <xs:complexType name="ActionFilter">...</xs:complexType>

  <xs:complexType name="Label"/>

  <xs:complexType name="DateTimeFilter">...</xs:complexType>

  <xs:complexType name="StringFilterNVa">...</xs:complexType>

  <xs:complexType name="StringFilterNVa">...</xs:complexType>

  <xs:complexType name="StringFilterNVa">...</xs:complexType>

  <xs:complexType name="StringFilterNTe">...</xs:complexType>

  <xs:element name="ActivityRecordS">...</xs:element>

</xs:schema>
```

Review the following for additional information:

- Filters
- Operators

9.2.2. Example

XML

```
<?xml version="1.0" encoding="utf-8"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <FilterList>
    <Who Operator="NotEqualTo">Administrator</Who>
    <AuditedSystem>Active Directory</AuditedSystem>
    <AuditedSystem Operator="StartsWith">Exchange</AuditedSystem>
    <Action>Removed</Action>
    <Action>Added</Action>
    <ObjectType Operator="DoesNotContain">Group</ObjectType>
    <When>
      <From>2015-01-16T16:30:00+11:00</From>
      <To>2016-01-01T00:00:00Z</To>
    </When>
  </FilterList>
</ActivityRecordSearch>
```

JSON

```
{
  "FilterList": {
    "Who": { "NotEqualTo": "Administrator" },
    "AuditedSystem": [ "Active Directory", { "StartsWith": "Exchange" } ],
    "Action": [ "Added", "Removed" ],
    "ObjectType": { "DoesNotContain": "Group" },
    "When": {
      "From": "2015-01-16T16:30:00+11:00",
      "To": "2016-01-01T00:00:00Z"
    }
  }
}
```

9.2.3. Reference for Creating Search Parameters File

Review this section to learn more about operators and how to apply them to Activity Record filters to create a unique search. You can:

- Add different filters to your search. Search results will be sorted by all selected filters since they work as a logical AND.

Format	Example

XML	`<Who Operator="Equals">Admin</Who>`
	`<AuditedSystem Operator="NotEqualTo">Active Directory</AuditedSystem>`
	`<What>User</What>`

JSON	`"Who" : { "Equals" : "Admin" },`
	`"AuditedSystem" : { "NotEqualTo" : "Active Directory" },`
	`"What" : "User"`

- Specify several values for the same filter. To do this, add two entries one after another.

  Entries with **Equals**, **Contains**, **StartsWith**, and **EndsWith** operators work as a logical OR (Activity Records with either of following values will be returned). Entries with **DoesNotContain** and **NotEqualTo** operators work as a logical AND (Activity Records with neither of the following values will be returned).

Format	Example
XML	`<Who>Admin</Who>` `<Who>Analyst</Who>`
JSON	`"Who" : [ "Admin" , "Analyst" ]`

  NOTE: Use square brackets to add several values for the entry.

Review the following for additional information:

- Filters
- Operators

The table below shows filters and Activity Records matching them.

Filters	Matching Activity Records

• XML:
`<Who>Administrator</Who>`
`<AuditedSystem>`
`SharePoint`
`</AuditedSystem>`
`<Action Operator="NotEqualTo">`
`Read`
`</Action>`
• JSON:
	Retrieves all Activity Records where administrator made any actions on SharePoint, except Read.
	• XML:
	`<ActivityRecord>`
	`<Action>Added</Action>`
	`<AuditedSystem>SharePoint</AuditedSystem>`
	`<ObjectType>List</ObjectType>`
	`<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>`
	`<What>http://demolabsp/lists/Taskslist</What>`
	`<When>2016-02-17T09:28:35Z</When>`

Filters	Matching Activity Records

"Who" : "Admin",
"AuditedSystem" : "SharePoint",
"Action" : {
"NotEqualTo" : "Read"
}
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Removed</Action>
	<AuditedSystem>SharePoint</AuditedSystem>
	<ObjectType>List</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D15857</RID>
	<What>http://demolabsp/lists/Old/Taskslist</What>
	<When>2016-02-17T09:28:35Z</When>
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	• JSON:
	{
	"Action": "Added",
	"AuditedSystem": "SharePoint",
	"ObjectType": "List",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What": "http://demolabsp/lists/Taskslist",
	"When": "2016-02-17T09:28:35Z",
	"Where": "http://demolabsp",
	"Who": "Enterprise\\Administrator",
	"Workstation": "172.28.15.126"
	},
	{
	"Action": "Removed",
	"AuditedSystem": "SharePoint",
	"ObjectType": "List",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D15857",
	"What": "http://demolabsp/lists/Old/Taskslist",
	"When": "2016-02-17T09:28:35Z",
	"Where": "http://demolabsp",
	"Who": "Enterprise\\Administrator",
	"Workstation": "172.28.15.126"
	}

• XML: <Who>Administrator</Who> <Action>Added</Action> • JSON: "Who" : "Administrator", "Action" : "Added"	Retrieves all Activity Records where administrator added an object within any audited system. • XML: <ActivityRecord> <Action>Added</Action> <AuditedSystem>SharePoint</AuditedSystem> <ObjectType>List</ObjectType>

Filters	Matching Activity Records

	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
	<What>http://demolabsp/lists/Taskslist</What>
	<When>2016-02-17T09:28:35Z</When>
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Added</Action>
	<AuditedSystem>Exchange</AuditedSystem>
	<ObjectType>Mailbox</ObjectType>
	<RID>20160211163547592O7E9DDCEEB674986AD30CD3D13F5DEA3</RID>
	<What>Shared Mailbox</What>
	<When>2016-02-10T14:46:00Z</When>
	<Where>eswks.enterprise.local</Where>
	<Who>Enterprise\Administrator</Who>
	</ActivityRecord>
	• JSON:
	{
	"Action": "Added",
	"AuditedSystem": "SharePoint",
	"ObjectType": "List",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What": "http://demolabsp/lists/Taskslist",
	"When": "2016-02-17T09:28:35Z",
	"Where": "http://demolabsp",
	"Who": "Enterprise\\Administrator",
	"Workstation": "172.28.15.126"
	},
	{
	"Action": "Added",
	"AuditedSystem": "Exchange",
	"ObjectType": "Mailbox",
	"RID": "20160211163547592O7E9DDCEEB674986AD30CD3D13F5DEA3",
	"What": "Shared Mailbox",
	"When": "2016-02-10T14:46:00Z",
	"Where": "eswks.enterprise.local",
	"Who": "Enterprise\\Administrator"
	}

• XML: <Who>Admin</Who> <Who>Analyst</Who> • JSON: "Who" : ["Admin" , "Analyst"]	Retrieves all Activity Records where admin or analyst made any changes within any audited system. • XML: <ActivityRecord> <Action>Added</Action> <AuditedSystem>File Servers</AuditedSystem>

Filters	Matching Activity Records

	<ObjectType>Folder</ObjectType>
	<RID>2016021116354759207E9DDCEEB674986AD30CD3D13F5DDA3</RID>
	<What>Annual_Reports</What>
	<When>2016-02-10T14:46:00Z</When>
	<Where>wks.enterprise.local</Where>
	<Who>Enterprise\Admin</Who>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Removed</Action>
	<AuditedSystem>Active Directory</AuditedSystem>
	<ObjectType>User</ObjectType>
	<RID>2016021116354759207E9DDCEEB674986AD30CD3D13F5DAA3</RID>
	<What>Anna.Smith</What>
	<When>2016-02-10T10:46:00Z</When>
	<Where>dc1.enterprise.local</Where>
	<Who>Enterprise\Analyst</Who>
	<Workstation>172.28.6.15</Workstation>
	</ActivityRecord>
	• JSON:
	{
	"Action": "Added",
	"AuditedSystem": "File Servers",
	"ObjectType": "Folder",
	"RID": "2016021116354759207E9DDCEEB674986AD30CD3D13F5DDA3",
	"What": "Annual_Reports",
	"When": "2016-02-10T14:46:00Z",
	"Where": "wks.enterprise.local",
	"Who": "Enterprise\\Admin"
	},
	{
	"Action": "Removed",
	"AuditedSystem": "Active Directory",
	"ObjectType": "User",
	"RID": "2016021116354759207E9DDCEEB674986AD30CD3D13F5DAA3",
	"What": "Anna.Smith",
	"When": "2016-02-10T10:46:00Z",
	"Where": "dc1.enterprise.local",
	"Who": "Enterprise\\Analyst",
	"Workstation": "172.28.6.15"
	}

• XML: <When> <LastSevenDays/> </When> <When> <From>	Retrieves all Activity Records for all audited systems and users within a specified data range: • January 16, 2016 — February 1, 2016 • March 11, 2016 — March 17, 2016 (assume, today is

Filters	Matching Activity Records

```
    2016-01-16T16:30:00Z
  </From>
  <To>
    2016-02-01T00:00:00Z
  </To>
</When>
```

- JSON:

```
"When" : [
  "LastSevenDays",
  {
    "From" : "2016-01-16T16:30:00Z",
    "To" : "2016-02-01T00:00:00Z"
  }
]
```

March, 17).

- XML:

```
<ActivityRecord>
  <Action>Modified</Action>
  <AuditedSystem>Exchange Online</AuditedSystem>
  <ObjectType>Mailbox</ObjectType>
  <RID>20160217093959797097D56DDA034420B9044249CC15EC5A</RID>
  <What>Shared Mailbox</What>
  <When>2016-03-17T09:37:11Z</When>
  <Where>BLUPR05MB1940</Where>
  <Who>admin@enterprise.onmicrosoft.com</Who>
</ActivityRecord>
<ActivityRecord>
  <Action>Successful Logon</Action>
  <AuditedSystem>Logon Activity</AuditedSystem>
  <ObjectType>Logon</ObjectType>
  <RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
  <What>stationexchange.enterprise.local</What>
  <When>2016-02-17T09:28:35Z</When>
  <Where>enterprisedc1.enterprise.local</Where>
  <Who>ENTERPRISE\Administrator</Who>
  <Workstation>stwin12R2.enterprise.local</Workstation>
</ActivityRecord>
```

- JSON:

```
{
  "Action": "Modified",
  "AuditedSystem": "Exchange Online",
  "ObjectType": "Mailbox",
  "RID": "20160217093959797097D56DDA034420B9044249CC15EC5A",
  "What": "Shared Mailbox",
  "When": "2016-03-17T09:37:11Z",
  "Where": "BLUPR05MB1940",
  "Who": "admin@enterprise.onmicrosoft.com"
},
{
  "Action": "Successful Logon",
  "AuditedSystem": "Logon Activity",
  "ObjectType": "Logon",
  "RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
  "What": "stationexchange.enterprise.local",
  "When": "2016-02-17T09:28:35Z",
  "Where": "enterprisedc1.enterprise.local",
  "Who": "ENTERPRISE\\Administrator",
  "Workstation": "stwin12R2.enterprise.local"
```

Filters	Matching Activity Records

	}
	- XML:
	Retrieves all Activity Records for Logon Activity audited system irrespective of who made the change and when it was made.
	- XML:
	<ActivityRecord>
	<Action>Successful Logon</Action>
	<AuditedSystem>Logon Activity</AuditedSystem>
	<ObjectType>Logon</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
	<What>stationexchange.enterprise.local</What>
	<When>2016-02-17T09:28:35Z</When>
	<Where>enterprisedc1.enterprise.local</Where>
	<Who>ENTERPRISE\Administrator</Who>
	<Workstation>stwin12R2.enterprise.local</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Successful Logon</Action>
	<AuditedSystem>Logon Activity</AuditedSystem>
	<ObjectType>Logon</ObjectType>
	<RID>20160217093959797097D56DDA034420B9044249CC15EC5A</RID>
	<What>stationwin12r2.enterprise.local</What>
	<When>2016-02-17T09:37:11Z</When>
	<Where>enterprisedc2.enterprise.local</Where>
	<Who>ENTERPRISE\Analyst</Who>
	<Workstation>stwin12R2.enterprise.local</Workstation>
	</ActivityRecord>
	- JSON:
	{
	"Action": "Successful Logon",
	"AuditedSystem": "Logon Activity",
	"ObjectType": "Logon",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What": "stationexchange.enterprise.local",
	"When": "2016-02-17T09:28:35Z",
	"Where": "enterprisedc1.enterprise.local",
	"Who": "ENTERPRISE\\Administrator",
	"Workstation": "stwin12R2.enterprise.local"
	},
	{
	"Action": "Successful Logon",
	"AuditedSystem": "Logon Activity",
	"ObjectType": "Logon",
	"RID": "20160217093959797097D56DDA034420B9044249CC15EC5A",
	"What": "stationwin12r2.enterprise.local",

- XML: <AuditedSystem> Logon Activity </AuditedSystem> - JSON: "Audited System" : "Logon Activity"	

Filters	Matching Activity Records

	<pre>"When": "2016-02-17T09:37:11Z",
	"Where": "enterprisedc2.enterprise.local",
	"Who": "ENTERPRISE\\Analyst",
	"Workstation": "stwin12R2.enterprise.local"
	}</pre>

9.2.3.1. Filters

Review the table below to learn more about filters. The filters correspond to Activity Record fields.

Filter Name	Filter Description	Supported Operators
RID	Limits your search to a unique key of the Activity Record. Max length: 49.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Who	Limits your search to a specific user who made the change (e.g., *Enterprise\ Administrator*). Max length: 255.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Where	Limits your search to a resource where the change was made (e.g., *Enterprise-SQL*, *FileStorage.enterprise.local*). The resource name can be a FQDN or NETBIOS server name, Active Directory domain or container, SQL Server instance, SharePoint farm, VMware host, etc. Max length: 255.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
ObjectType	Limits your search to objects of a specific type only (e.g., *user*). Max length: 255.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith

Filter Name	Filter Description	Supported Operators
		• EndsWith
What	Limits your search to a specific object that was changed (e.g., *NewPolicy*) . Max length: 1073741822.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
AuditedSystem	Limits your search to the selected audited system only (e.g., *Active Directory*). Max length: 1073741822.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Workstation	Limits your search to an originating workstation from which the change was made (e.g., *WKSwin12.enterprise.local*). Max length: 1073741822.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Detail	Limits your search results to entries that contain the specified information in **Detail**. Normally contains data specific to your audited system, e.g., assigned permissions, before and after values, start and end dates. This filter can be helpful when you are looking for a unique entry. Max length: 1073741822.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Before	Limits your search results to entries that contain the specified before value in **Detail**. Max length: 536870911.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith

Filter Name	Filter Description	Supported Operators
After	Limits your search results to entries that contain the specified after value in the **Detail**. Max length: 536870911.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Action	Limits your search results to certain actions: • Added • Removed • Modified • Read • Moved • Renamed • Checked in • Discard check out • Successful Logon • Copied • Session start • Activated • Add (Failed Attempt) • Remove (Failed Attempt) • Modify (Failed Attempt) • Read (Failed Attempt) • Move (Failed Attempt) • Rename (Failed Attempt) • Checked out • Logoff • Failed Logon • Sent • Session end	• Equals (default) • NotEqualTo
When	Limits your search to a specified time range. Netwrix allows defining the **When** filter in two ways simultaneously. You can select a timeframe modifier (one of the enumerated values) for the **When** and values in the **To** and **From**. **To** and **From** support the following date time formats: • **YYYY-mm-ddTHH:MM:SSZ**—Indicates UTC time (zero offset) • **YYYY-mm-ddTHH:MM:SS+HH:MM**—Indicates time zones ahead of UTC (positive offset) • **YYYY-mm-ddTHH:MM:SS-HH:MM**—Indicates time zones behind UTC (negative offset)	1. Within timeframe: • Today • Yesterday • LastSevenDays • LastThrityDays 2. "From..To" interval

9.2.3.2. Operators

Review the table below to learn more about operators.

Operator	Description	Example
Contains	This broad match operator shows all entries that include a value specified in the filter.	Set the **Who** filter to **contains** *John*, to get the following results: *Domain1\John*, *Domain1\Johnson*, *Domain2\Johnny*.
Equals	This exact match operator shows all entries with the exact value specified. Make sure to provide a full object name or path.	Use this operator if you want to get precise results, e.g., *\\FS\Share\NewPolicy.docx*.
NotEqualTo	This negative exact match operator shows all entries except those with the exact value specified.	Set the **Who** filter to **NotEqualTo** *Domain1\John* to exclude the exact user specified and find all changes performed by other users, e.g., *Domain1\Johnson, Domain2\John*.
StartsWith	This operator shows all entries that start with the exact value specified.	Set the **Who** filter to **StartsWith** *Domain1\John* to find all changes performed by *Domain1\John*, *Domain1\Johnson*, and *Domain1\Johnny*.
EndsWith	This operator shows all entries that end with the exact value specified.	Set the **Who** filter to **EndsWith** *John* to find all changes performed by *Domain1\John*, *Domain2\Dr.John*, *Domain3\John*.
DoesNotContain	This negative broad match operator shows all entries except those that contain the value specified.	Set the **Who** filter to **DoesNotContain** *John* to exclude the following users, *Domain1\John*, *Domain2\Johnson*, and *Domain3\Johnny*.

9.3. Activity Records

In Netwrix terms, one operable chunk of information is called the Activity Record. Netwrix Auditor Integration API processes both XML and JSON Activity Records. The Activity Records have the format similar to the following—the exact schema depends on operation (input or output).

Format	Example
XML	`<?xml version="1.0" encoding="UTF-8" ?>` `<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">`

Format	Example

```
 <ActivityRecord>
  <Who>Who</Who>
  <ObjectType>Object Type</ObjectType>
  <Action>Action</Action>
  <What>What</What>
  <When>When</When>
  <Where>Where</Where>
  <AuditedSystem>Audited System</AuditedSystem>
  <DetailList>
   <Detail>
    <Before>Before Value</Before>
    <After>After Value</After>
    <PropertyName>Property</PropertyName>
   </Detail>
  </DetailList>
 </ActivityRecord>
 <ActivityRecord>...</ActivityRecord>
</ActivityRecordList>
```

JSON

```
[
 {
  "Action": "Action",
  "AuditedSystem": "Audited System",
  "DetailList": [
   {
    "Before": "Before Value",
    "After": "After Value",
    "PropertyName": "Property"
   }
  ],
  "ObjectType": "Object Type",
  "What": "What",
  "When": "When",
  "Where": "Where",
  "Who": "Who"
 },
 {...}
]
```

To feed data from a custom audit source to Netwrix Auditor, send a POST request containing Activity Records. See Write Activity Records for more information.

9.3.1. Schema

The Activity Records you want to feed to Netwrix Auditor must be compatible with input schema. The output schema resembles the input schema and can be used to validate Activity Records returned

by Netwrix Auditor before further data parsing.

Format	Schema description

XML	The file must be compatible with the XML schema. On computer where Netwrix Auditor Administrator Console is installed, you can find XSD file under *Netwrix_Auditor_ installation_ folder\Audit Core\API Schemas*.
	
	The `ActivityRecordList` root element includes the `ActivityRecord` elements. Each `ActivityRecord` contains values in the `Who`, `When`, `Where`, etc. fields. The `DetailList` element is not mandatory, it may include one or more `Detail` entries. The `Detail` element may contain sub-elements with values (e.g., before and after values). For input Activity Records, the audited system is automatically set to **Netwrix API**.
	
	NOTE: **minOccurs="0"** indicates that element is optional and may be absent in the Search parameters.
	
	`<?xml version="1.0" encoding="utf-8"?>`
	`<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"`
	`targetNamespace="http://schemas.netwrix.com/api/v1/activity_records/"`
	`xmlns="http://schemas.netwrix.com/api/v1/activity_records/"`
	`elementFormDefault="qualified">`
	
	`<xs:complexType name="StringFilter">...</xs:complexType>`
	
	`<xs:simpleType name="ActionEnum">...</xs:simpleType>`
	
	`<xs:complexType name="ActionFilter">...</xs:complexType>`
	
	`<xs:complexType name="Label"/>`
	
	`<xs:complexType name="DateTimeFilter">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNTe">...</xs:complexType>`
	
	`<xs:element name="ActivityRecordS">...</xs:element>`
	
	`</xs:schema>`

JSON	Activity Records are sent as an array collected within square brackets [ ]. Each `ActivityRecord` object is collected in braces {} and contains values in the `Who`, `When`, `Where`, etc. fields. The `DetailList` field is not mandatory, it may include one or more detail. The `Detail` field may contain sub-fields with values (e.g., before and after values). For input Activity Records, the audited system is automatically set to **Netwrix API**.

9.3.2. Example

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ActivityRecord>
    <Action>Modified</Action>
    <ObjectType>Mailbox</ObjectType>
    <What>Shared Mailbox</What>
    <When>2016-03-17T09:37:11Z</When>
    <Where>BLUPR05MB1940</Where>
    <Who>admin@enterprise.onmicrosoft.com</Who>
    <DetailList>
      <Detail>
        <Before>1</Before>
        <After>2</After>
        <PropertyName>Custom_attribute</PropertyName>
      </Detail>
    </DetailList>
  </ActivityRecord>
</ActivityRecordList>
```

JSON

```
[
  {
    "Action": "Modified",
    "ObjectType": "Mailbox",
    "What": "Shared Mailbox",
    "When": "2016-03-17T09:37:11Z",
    "Where": "BLUPR05MB1940",
    "Who": "admin@enterprise.onmicrosoft.com",
    "DetailList": [
      {
        "PropertyName": "Custom_Attribute",
        "Before" : "1",
        "After" : "2"
      }
    ]
  }
]
```

9.3.3. Reference for Creating Activity Records

The table below describes Activity Record elements.

NOTE: Netwrix recommends limiting the input Activity Records file to 50MB and maximum 1,000 Activity Records.

Element	Mandatory	Datatype	Description
Activity Record main elements			
RID	No	string	RID is a unique key of the Activity Record. The identifier is created automatically when you write an Activity Record to the Audit Database. RID is included in output Activity Records only.
Who	Yes	nvarchar 255	A specific user who made the change (e.g., *Enterprise\Administrator*).
Action	Yes	—	Activity captured by Netwrix Auditor (varies depending on the audited system): • Added • Removed • Modified • Read • Moved • Renamed • Checked in • Discard check out • Successful Logon • Copied • Session start • Activated • Add (Failed Attempt) • Remove (Failed Attempt) • Modify (Failed Attempt) • Read (Failed Attempt) • Move (Failed Attempt) • Rename (Failed Attempt) • Checked out • Logoff • Failed Logon • Sent • Session end
What	Yes	nvarchar max	A specific object that was changed (e.g., *NewPolicy*).
When	Yes	dateTime	The moment when the change occurred. **When** supports the following datetime formats: • **YYYY-mm-ddTHH:MM:SSZ**—Indicates UTC time (zero offset) • **YYYY-mm-ddTHH:MM:SS+HH:MM**—Indicates time zones ahead of UTC (positive offset) • **YYYY-mm-ddTHH:MM:SS-HH:MM**—Indicates time zones behind UTC (negative offset)

Element	Mandatory	Datatype	Description
Where	Yes	nvarchar 255	A resource where the change was made (e.g., *Enterprise-SQL*, *FileStorage.enterprise.local*). The resource name can be a FQDN or NETBIOS server name, Active Directory domain or container, SQL Server instance, SharePoint farm, VMware host, etc.
ObjectType	Yes	nvarchar 255	An type of affected object or its class (e.g., *user, mailbox*).
Audited System	No	string	IT infrastructure monitored with Netwrix Auditor (e.g., *Active Directory*). For input Activity Records, the audited system is automatically set to **Netwrix API**.
Workstation	No	nvarchar max	An originating workstation from which the change was made (e.g., *WKSwin12.enterprise.local*).
DetailList	No	—	Information specific to the audited system, e.g., assigned permissions, before and after values, start and end dates. References details.
IsArchiveOnly	No	—	**IsArchiveOnly** allows to save Activity Record to the Long-Term Archive only. In this case, these Activity Records will not be available for search in the Netwrix Auditor client.
Detail elements (provided that DetailList exists)			
PropertyName	Yes	nvarchar 255	The name of a modified property.
Before	No	ntext	The previous value of the modified property.
After	No	ntext	The new value of the modified property.

10. Response Status Codes

Code	Status	Write Activity Records	Retrieve, search Activity Records
200 OK	Success	Success. The body is empty. Activity Records were written to the Audit Database and the Long-Term Archive.	Success. The body contains Activity Records. Activity Records were retrieved from the Audit Database.
400 Bad Request	Error	Error validating Activity Records. Make sure the Activity Records are compatible with Activity Records	Error validating request parameters or post data. Make sure the post data files (Continuation mark, Search parameters) are compatible with their schemas and the `?count=` parameter is valid.
401 Unauthorized	Error	The request is unauthorized. The body is empty. See Authentication for more information.	
404 Not Found	Error	Error addressing the endpoint. The body is empty. The requested endpoint does not exist (e.g., /netwrix/api/v1/mynewendpoint/).	
405 Method Not Allowed	Error	Error addressing the endpoint. The body is empty. Wrong HTTP request was sent (any except POST).	Error addressing the endpoint. The body is empty. Wrong HTTP request was sent (any except GET or POST).
413 Request Entity Too Large	Error	Error transferring files. The body is empty. The posted file exceeds supported size.	
500 Internal Server Error	Error	Error writing Activity Records to the Audit Database or the Long-Term Archive: • One or more Activity Records were not processed. • Netwrix Auditor has expired. • Internal error occurred.	Error retrieving Activity Records from the Audit Database: • Netwrix Auditor has expired. • The **Netwrix Auditor Archive Service** is unreachable. Try restarting the service on the computer that hosts Netwrix Auditor Server.

Code	Status	Write Activity Records	Retrieve, search Activity Records
			• Internal error occurred.
503 Service Unavailable	Error	The **Netwrix Auditor Archive Service** is busy or unreachable. Try restarting the service on the computer that hosts Netwrix Auditor Server.	—

NOTE: Most failed requests contain error in the response body (except those with empty body, e.g., 404, 405). See Error Details for more information.

10.1. Error Details

On error, most requests contain an error description in the response body (except some requests with empty body, e.g., 404, 405). See Response Status Codes for more information.

The error details include:

Block	Description
Category	Defines the type of error (XML formatting-related error, invalid input-related error, etc.)
Description	Provides details about this error.
Location	(optional) Provides a link to a corrupted text in request. **NOTE:** XML is considered a default format for Netwrix Auditor Integration API. Error location is defined in XML format.

The error details have the format similar to the following:

Format	Example

XML	`<?xml version="1.0" encoding="UTF-8" ?>`
	`<ErrorList xmlns="http://schemas.netwrix.com/api/v1/">`
	`  <Error>`
	`  <Category>Category</Category>`
	`  <Description>Error Description</Description>`
	`  <Location>Error Location</Location>`
	`  </Error>`
	`</ErrorList>`

Format	Example

JSON	<pre>{
	  "ErrorList": [
	  {
	  "Category": "Category",
	  "Description": "Error Description",
	  "Location": "Error Location"
	  }
	  ]
	}</pre>

Review examples below to see how error details correspond to invalid requests.

Request	Error details returned

Invalid request:
 	
XML:
 	
<pre>curl -H "Content-Type: application/xml; Charset=UTF-8" https://WKSWin12R2:9699/netwrix/api/v1/activity_records/search -u Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Search.xml</pre>
<pre><?xml version="1.0" encoding="utf-8"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <FilterList>
  <Who>Administrator</Who>
  <AuditedSystem>Active Directory
  <Action>Modified</Action>
  </FilterList>
</ActivityRecordSearch></pre>
• JSON:
 	
<pre>curl -H "Content-Type: application/json; Charset=UTF-8" https://WKSWin12R2:9699/netwrix/api/v1/activity_records/search?format=json -u Enterprise\NetwrixUser:NetwrixIsCool --data-binary @C:\APIdocs\Search.xml</pre>
<pre>{
  "FilterList": {
  "Who": "Administrator",
  "AuditedSystem": "Active Directory</pre>
	400 Bad Request
	
	• XML:
	
	<pre><?xml version="1.0" encoding="UTF-8" ?>
	<ErrorList xmlns="http://schemas.netwrix.com/api/v1/">
	  <Error>
	  <Category>XMLError</Category>
	  <Description>0xC00CE56D End tag 'FilterList'
	  does not match the start tag 'AuditedSystem'
	  </Description>
	  </Error>
	</ErrorList></pre>
	• JSON:
	
	NOTE: If JSON is corrupted, server returns 500 Internal Server Error with empty body.

Request	Error details returned

<pre>"Action": "Added"
}
}</pre>	

Invalid request:
• XML:
<pre>curl https://WKSWin12R2:9699/
netwrix/api/v1/activity_records/
enum?count=**FIVE** -u Enterprise\
NetwrixUser:NetwrixIsCool</pre>• JSON:
<pre>curl https://WKSWin12R2:9699/
netwrix/api/v1/activity_records/
enum?format=json&count=**FIVE** -u
Enterprise\NetwrixUser:
NetwrixIsCool</pre>
	400 Bad Request
	• XML:
	<pre><?xml version="1.0" encoding="UTF-8" ?>
	<ErrorList xmlns="http://schemas.netwrix.com/api/v1/">
	<Error>
	<Category>InputError</Category>
	<Description>Invalid count parameter specified.
	Error details: 0x80040204 Cannot convert the
	attribute data type
	</Description>
	</Error>
	</ErrorList></pre>• JSON:
	<pre>{
	"ErrorList": [
	{
	"Category": "InputError",
	"Description": "Invalid count parameter specified.
	Error details: 0x80040204 Cannot convert the
	attribute data type"
	}
	]
	}</pre>

Valid request, but the Audit Database is unreachable:
• XML:
<pre>curl https://WKSWin12R2:9699/
netwrix/api/v1/activity_
records/enum -u Enterprise\
NetwrixUser:NetwrixIsCool</pre>• JSON:
<pre>curl https://WKSWin12R2:9699/
netwrix/api/v1/activity_
records/enum?format=json -u
Enterprise\NetwrixUser:
NetwrixIsCool</pre>
	500 Internal Server Error
	• XML:
	<pre><?xml version="1.0" encoding="UTF-8" ?>
	<ErrorList xmlns="http://schemas.netwrix.com/api/v1/">
	<Error>
	<Category>ServerError</Category>
	<Description>0x80040C0A SQL Server cannot be
	contacted, connection is lost (0x80040C0A SQL
	Server cannot be contacted, connection is lost
	(0x80004005 [DBNETLIB][ConnectionOpen (Connect()).
	]SQL Server does not exist or access denied.))
	[0x00007FFDCC06BBC8,0x00007FFDB99EF4BA;
	0x00007FFDB99BEEEF,0x00007FFDB99EF4DC]
	</Description>
	</Error>
	</ErrorList></pre>• JSON:

Request	Error details returned

	{
	"ErrorList": [
	{
	"Category": "ServerError",
	"Description": "0x80040C0A SQL Server cannot be contacted, connection is lost (0x80040C0A SQL Server cannot be contacted, connection is lost (0x80004005 [DBNETLIB][ConnectionOpen (Connect()).
	]SQL Server does not exist or access denied.))
	[0x00007FFDCC06BBC8,0x00007FFDB99EF4BA;
	0x00007FFDB99BEEEF,0x00007FFDB99EF4DC]"
	}
	]
	}

11. Add-Ons

The Netwrix Auditor Add-on Store provides free add-ons developed by Netwrix Corp. and your peers in the community. The add-ons help you leverage integration between your on-premises or cloud applications and Netwrix Auditor.

The list of available add-ons keeps growing because with the new RESTful API, the integration capabilities of Netwrix Auditor are unlimited. Netwrix encourages users to develop add-ons, upload them to Netwrix website, and share with community.

Currently, the following add-ons are available:

Add-on	Technology	Description
AWSTrailAddon	PowerShell	Exports user activity data from your Amazon Web Services using CloudTrail and feeds events to the Audit Database. Use this script if you want to get more out of native Amazon auditing.
CefExportAddon	PowerShell	Exports Activity Records from the Audit Database to a cef file. Use this script to integrate data collected by Netwrix Auditor with SIEM solutions that use CEF files as input data.
EventLogExportAddon	PowerShell	Exports Activity Records from the Audit Database to a custom Windows event log—Netwrix Auditor Activity. Use this script to integrate data collected by Netwrix Auditor with SIEM solutions that use events as input data.
HPArcSightAddon	PowerShell	Exports Activity Records from the Audit Database to HP ArcSight in its native CEF format. Use this script to integrate Netwrix Auditor and ArcSight and extend auditing possibilities.

Netwrix Auditor Integration API uses HTTPS with an automatically generated certificate for running requests to its endpoints. By default, add-ons are configured to accept all certificates that is appropriate for evaluation purposes and allows running the script without adjusting.

Nonetheless, Netwrix recommends changing the accept-all-certificates behavior by commenting out a corresponding string in the script (starts with `$WebRequest.ServerCertificate`). With enabled certificate validation, the first option is to continue using a Netwrix-generated certificate. The **Netwrix Integration API** certificate is created automatically along with Netwrix Auditor installation and is located in the **Personal** store. In this case, enable trust on the computer where you are going to run the script. The other option is to assign a new certificate acquired from any reliable source.

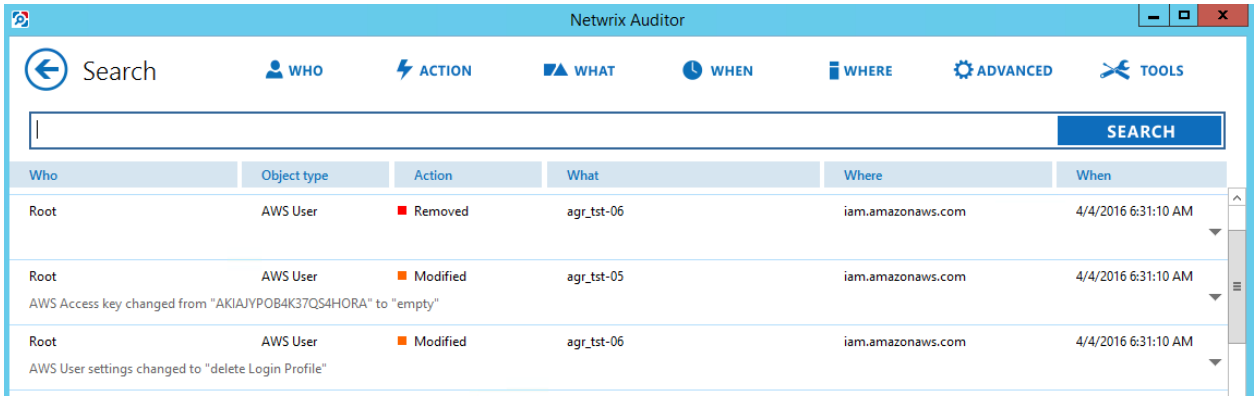
Refer to Security for detailed instructions on how to assign a new certificate and enable trust on remote computers.

To run a script

1. Download a script from Netwrix website.
2. Right-click a script and select **Edit**. **Windows PowerShell ISE** will start.
3. Review the script.

   NOTE: For AWSTrailAddon, provide mandatory parameters: AWS region endpoint, AWS access key ID, and AWS secret access key.

4. Click (**Run Script**).
5. During the script execution, you will be prompted to provide details, e.g., Netwrix Auditor Server host name, your credentials, etc.
6. Review the results. For example:



12. IIS Forwarding

NOTE: While you can configure forwarding from any web server, this guide covers IIS configuration procedure only.

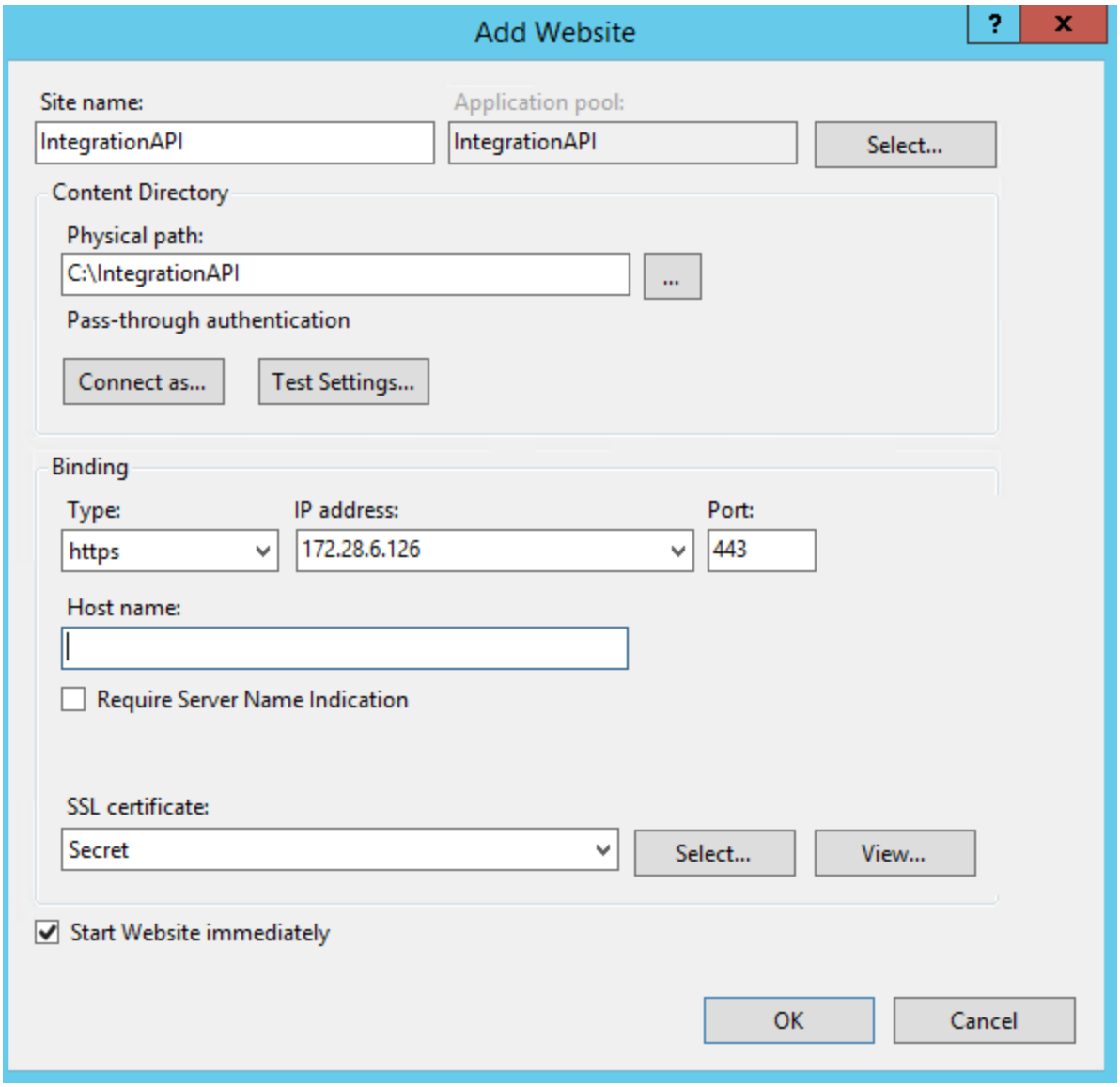
You can create a website in IIS and use it as a proxy for forwarding API requests. This is handy if for security reasons you do not want to make the Netwrix Auditor Server host name or address public. In this case, you can create a website with a short and user-friendly name and configure it to redirect requests to a server that hosts Netwrix Auditor Server and actually processes RESTful API requests. You can also configure authentication and authorization on IIS side.

For example, instead of addressing requests to `https://172.28.6.15:9699/netwrix/api/v1/activity_ records/enum` endpoint, you can send them to `https://enterprisewks/integrationAPI/activity_records/enum`.

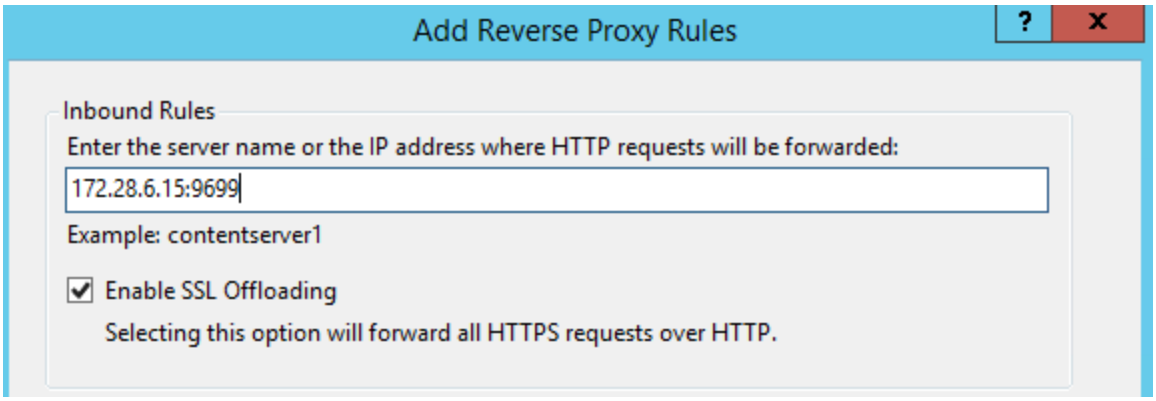
12.1. Configure IIS Forwarding

NOTE: The procedure below applies to IIS 8.5 integrated with Windows Server 2012 R2.

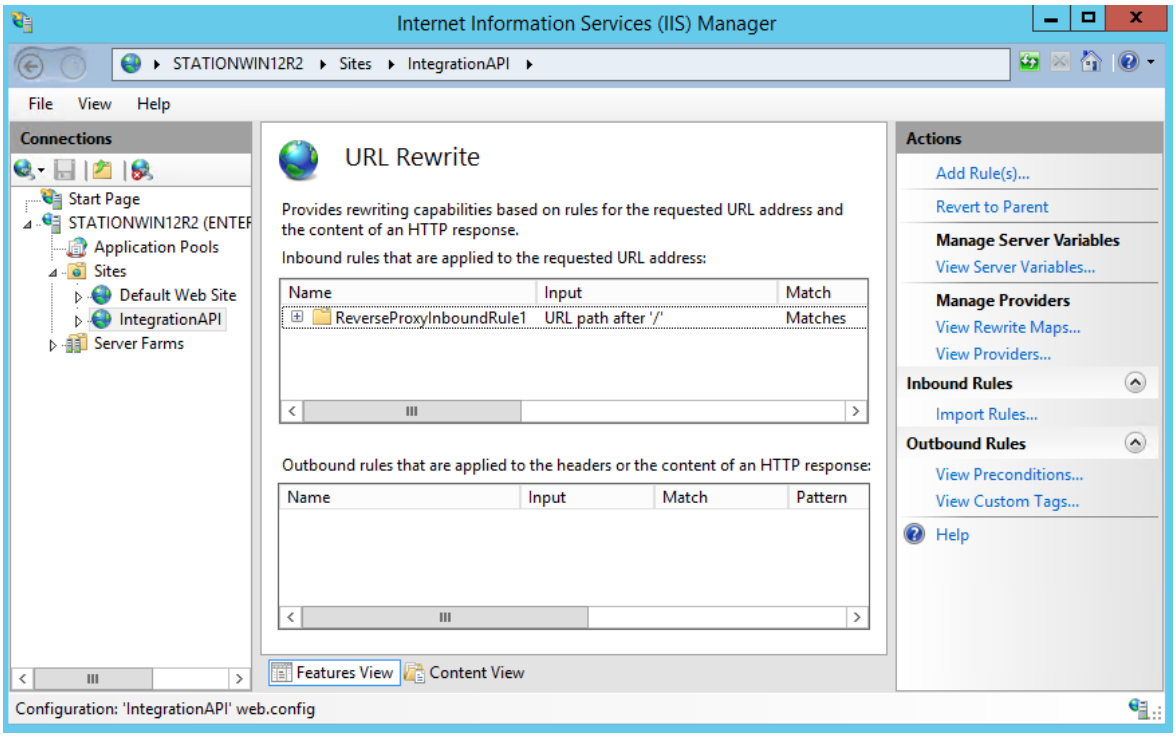
1. Make sure the **Web Server** role is installed on your server. Install the following components:
   - Application Request Routing
   - URL Rewrite
2. Create IIS website. To do this, navigate to **Start** → **Administrative Tools** → **Internet Information Services (IIS) Manager**. In the left, expand **your_computer_name** → **Sites** and select **Add Website** in the **Actions** pane. Create a website and configure authentication if necessary.



3. In your site settings, double-click **URL Rewrite** and select **Add Rule(s)**.
4. In the **Add Rule(s)** dialog, select **Reverse Proxy**. Select **OK** when prompted to enable **Application Request Routing** and proceed further.
5. In the **Add Reverse Proxy Rules** dialog that opens, provide a Netwrix Auditor Server host name or IP address.



6. Edit the newly created inbound rule.



7. On the **Edit Inbound Rule** page, complete the following fields and click **Apply**:

Option	Set to...
	Match URL
Requested URL	Matches the Pattern
Using	Regular Expressions
Pattern	activity_records/(.*) **NOTE:** In this case all requests containing "activity_records" will be

Option	Set to...
	forwarded. For example, *https://Enterprise/IntegrationAPI/activity_records/enum*.
Ignore case	Checked
	Action
Action type	Rewrite
Rewrite URL	https://*host:port*/netwrix/api/v1/activity_records/{R:1} where *host:port* is the name or IP address of the computer where Netwrix Auditor Server resides and port opened to communication. For example: *https://172.28.6.15:9699/netwrix/api/v1/activity_records/{R:1}*
Append query string	Checked
Log rewritten URL	Cleared
Stop processing of subsequent rules	Checked

Now you can send requests to your website that will forward them to proper Netwrix Auditor Integration API endpoints.

12.2. Usage Example—Forward Requests

The example below describes how to forward requests to another server.

1. Configure forwarding as described above.
2. Retrieve Activity Records from the Audit Database. See Retrieve Activity Records for more information.

Format	Request
XML	`curl https://172.28.15.126:80/integrationapi/activity_records/enum -u Enterprise\NetwrixUser:NetwrixIsCool`
JSON	`curl https://172.28.15.126:80/integrationapi/activity_records/enum?format=json -u Enterprise\NetwrixUser:NetwrixIsCool`

3. The request is automatically forwarded to endpoint starting with `https://172.28.6.15:9699/netwrix/api/v1/activity_records/`.

4. Receive the response. Below is an example of a successful GET request. The status is **200 OK**. For XML, a response body contains the `ActivityRecordList` root element with Activity Records and a Continuation mark inside. For JSON, a response body contains the `ActivityRecordList` array with Activity Records collected in braces {} and a Continuation mark.

XML

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA3MUI0NC0xRjk1LTQxRTAtOTE2Qi04RTU5NUU3MDJCMDh9
  IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY9Ij
  Q0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5RjlB
  Q0YxNzJFQyIvPjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <AuditedSystem>Active Directory</AuditedSystem>
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
    <Action>Added</Action>
    <When>2016-02-14T15:42:34Z</When>
    <Where>EnterpriseDC1.enterprise.local</Where>
    <Who>ENTERPRISE\Administrator</Who>
    <Workstation>EnterpriseDC1.enterprise.local</Workstation>
  </ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
  <ActivityRecord>...</ActivityRecord>
</ActivityRecordList>
```

JSON

```
{
  "ActivityRecordList": [
    {
      "Action": "Added",
      "AuditedSystem": "Active Directory",
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
      "When": "2016-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
```

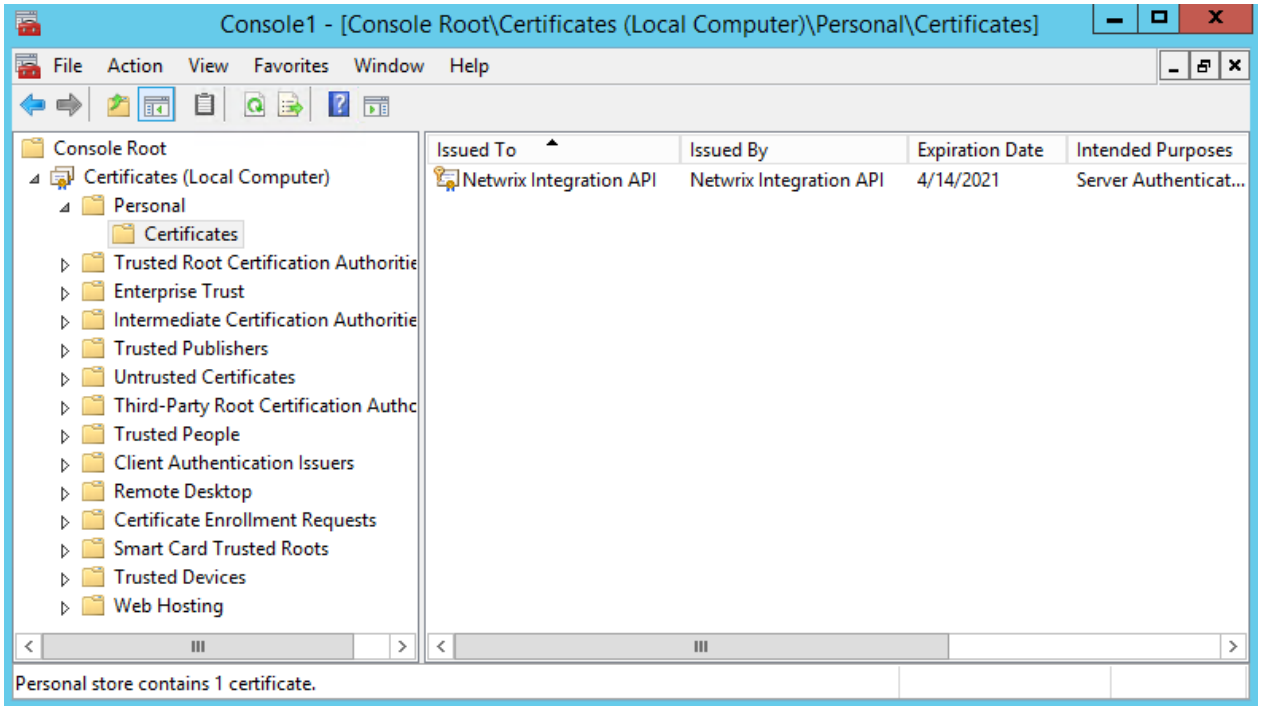
```
  ],
  "ContinuationMark": "PG5yPjxuIG49IntDRTBFNjNCQy1ENUVCLTQxQzgtQkE1Ni1BOTI4RjkxRjZCO
  DF9IiB0PSJDb250aW51YXRpb25NYXJrIj48YSBuPSJDb250aW51YXRpb25NYXJrIiB0PSIyNTgiPjx2IHY
  9IjQ0MzUxMzU2MTE5MDcwNzcyMjE6MjAxNjA0MDUxMDM3NDkxMjVGMUJFRkFBMzIyOTI0NDlCODREMzA5R
  jlBQ0YxNzJFQyIvPjwvYT48L24+PC9ucj4A"
}
```


5. Continue retrieving Activity Records. See Usage Example—Retrieve All Activity Records for more information.

13. Security

By default, Netwrix Auditor Integration API uses HTTPS for sending requests to its endpoints. Netwrix encrypts data with a self-signed automatically generated SSL certificate and strongly recommends you to replace it with a new secured certificate acquired from any reliable source.

The automatically generated **Netwrix Integration API** certificate is located in the **Personal** store. To enable trust on remote computers, install this certificate in the **Trusted Root Certification Authorities** store.



To manage API security settings with APIAdminTool.exe

Netwrix provides a command-line tool for managing Integration API. The tool allows switching between HTTP and HTTPS, assigning new certificates, etc.

1. On the computer where Netwrix Auditor Administrator Console is installed, start the **Command Prompt** and run the tool. The tool is located in the *Netwrix Auditor installation folder*, inside the *Audit Core* folder. For example:

   ```
   C:\>cd C:\Program Files (x86)\Netwrix Auditor\Audit Core

   C:\Program Files (x86)\Netwrix Auditor\Audit Core>APIAdminTool.exe
   ```

2. Execute one of the following commands depending on your task.

   NOTE: Review the tips for running the tool:

- Some commands require parameters. Provide parameters with values (parameter=value) if you want to use non-default. E.g., `APIAdminTool.exe api http port= 4431`.
- Append `help` to any command to see available parameters and sub-commands. E.g., `APIAdminTool.exe api help`.

To...	Execute...
Disable API	`APIAdminTool.exe api disable` **NOTE:** This command duplicates the checkbox on the **Integration API** page in Netwrix Auditor Administrator Console.
Switch to HTTP	`APIAdminTool.exe api http` **NOTE:** Netwrix recommends switching to HTTP only in safe intranet environments. To use a non-default port (9699), append a parameter port with value to the command above (e.g., `port= 4431`).
Switch to HTTPS	`APIAdminTool.exe api https` **NOTE:** Run this command if you want to continue using Netwrix-generated certificate. To use a non-default port (9699), append a parameter port with value to the command above (e.g., `port= 4431`).

Assign a new SSL certificate	`APIAdminTool.exe api https certificate`
	
	NOTE: Run this command if you want to apply a new certificate and use it instead default. You must add a certificate to the store before running this command.
	
	Provide parameters to specify a certificate:
	• For a certificate exported to a file:
	  • path—Mandatory, defines certificate location.
	  • store—Optional, defines the store name where certificate is located. By default, Personal.
	
	For example: `APIAdminTool.exe api https certificate path= C:\SecureCertificate.cef store= Personal`
	
	• For a self-signed certificate:
	  • subject—Mandatory, defines certificate name.

To...	Execute...
	• validFrom—Optional, defines a certificate start date. By default, today. • validTo—Optional, defines a certificate expiration date. By default, 5 years after a validFrom date. For example: `APIAdminTool.exe api https certificate subject= New validTo= 01/01/2021` • For a certificate specified using thumbprint: • store—Optional, defines the store name where certificate is located. By default, Personal. • thumbprint—Mandatory, defines a thumbprint identifier for a certificate. For example: `APIAdminTool.exe api https certificate thumbprint= 3478cda8586675e420511dc0fdf59078093eeeda`

Index