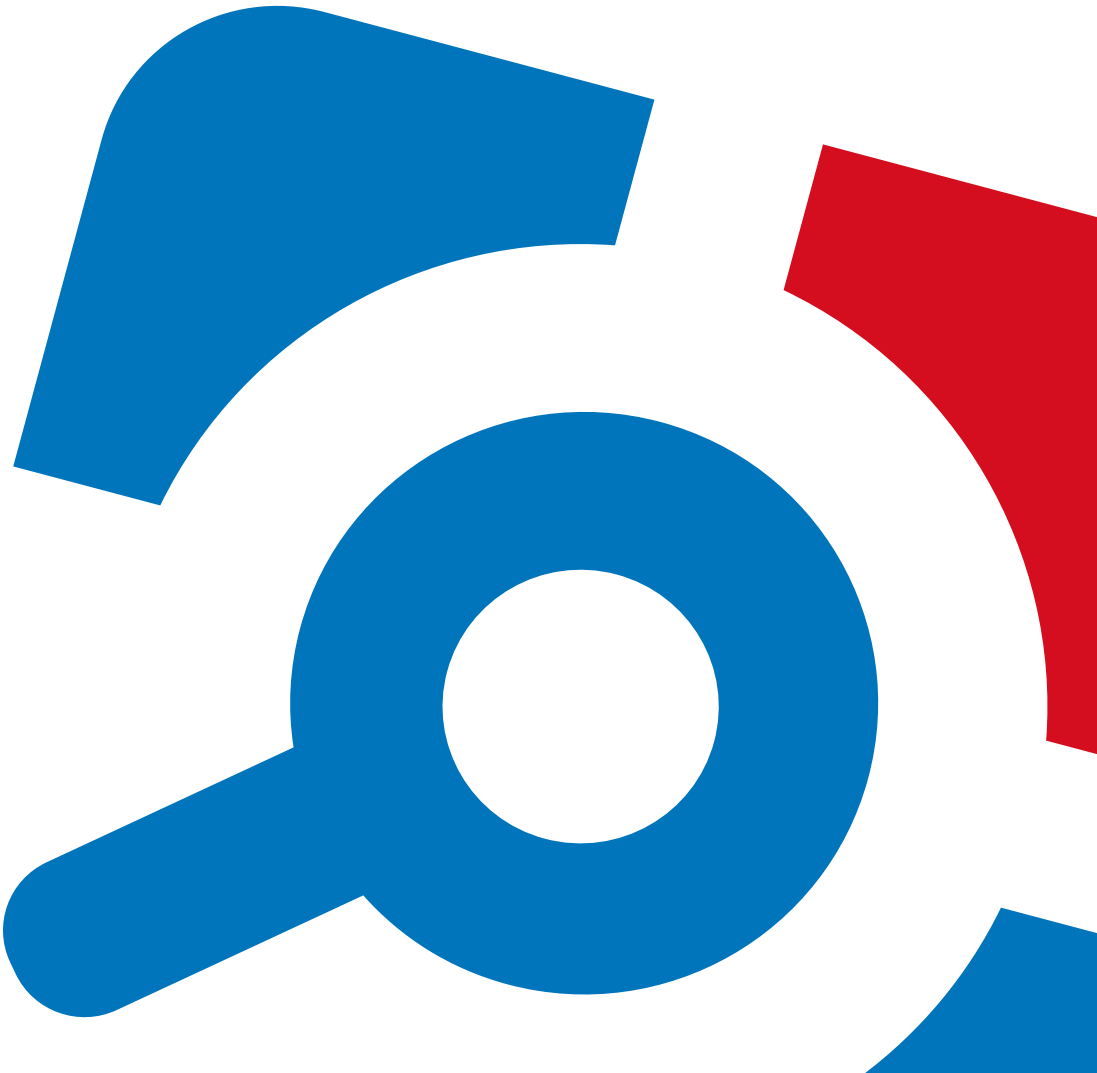
netwrix

Netwrix Auditor

Integration API Guide

Version: 9.5
12/4/2017



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

Looking for online version? Check out Netwrix Auditor help center.

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 a visibility platform for user behavior analysis and risk mitigation that enables control over changes, configurations and access in hybrid IT environments to protect data regardless of its location. The platform provides security analytics to detect anomalies in user behavior and investigate threat patterns before a data breach occurs.

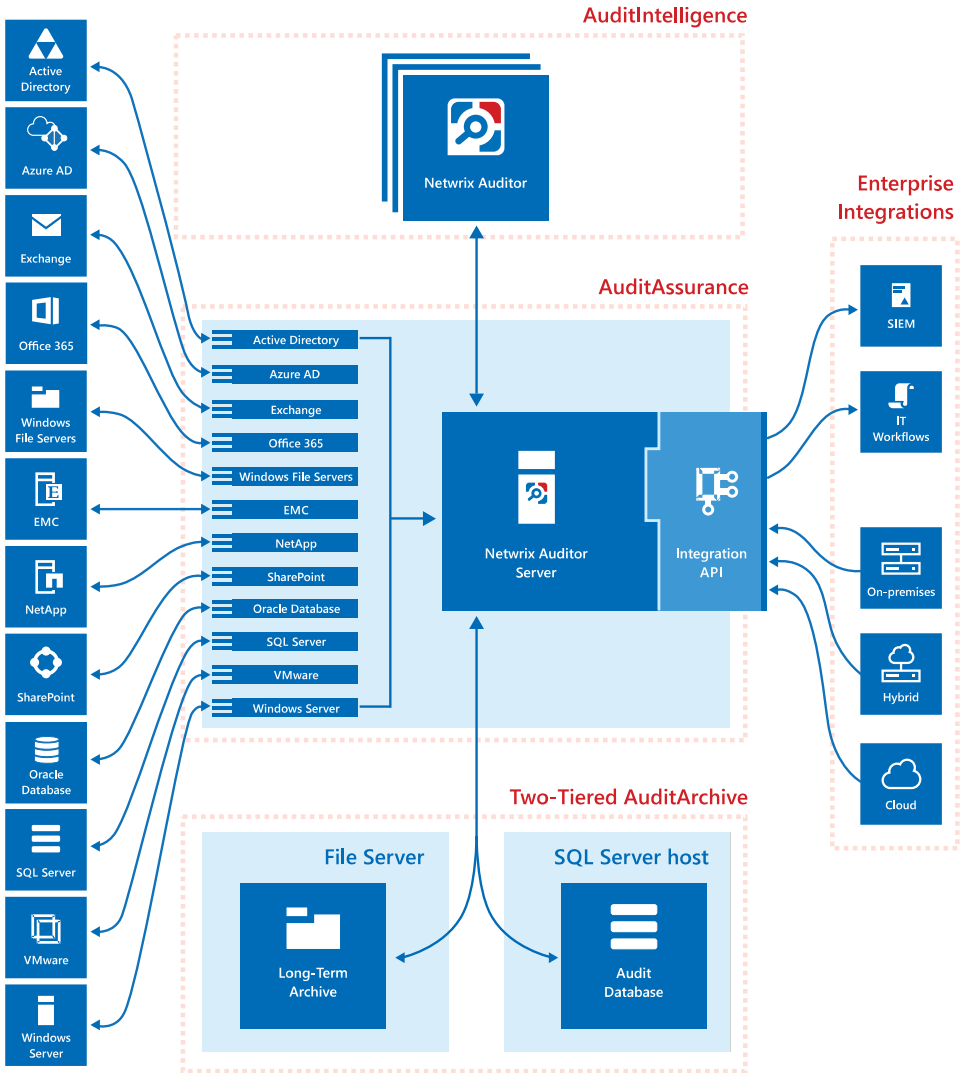
Netwrix Auditor includes applications for Active Directory, Azure AD, Exchange, Office 365, Windows file servers, EMC storage devices, NetApp filer appliances, SharePoint, Oracle Database, SQL Server, VMware, and Windows Server. Empowered with a RESTful API and user activity video recording, the platform delivers visibility and control across all of your on-premises or cloud-based IT systems in a unified way.

Major benefits:

- Detect insider threats—on premises and in the cloud
- Pass compliance audits with less effort and expense
- Increase productivity of IT security and operations teams

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, or simply **Intelligence**, is a brand new way of dealing with audit data, investigating incidents and enabling complete visibility across the entire IT infrastructure. **Intelligence** provides easy access to data and configuration for IT managers, business analysts and other relevant employees via a straightforward and user-friendly interface, **Netwrix Auditor client**. 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 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 data. It contains several internal components responsible for gathering data from data sources. **Integration API** is a RESTful API that

leverages data with custom on-premises or cloud systems even if they are not supported as data sources 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**

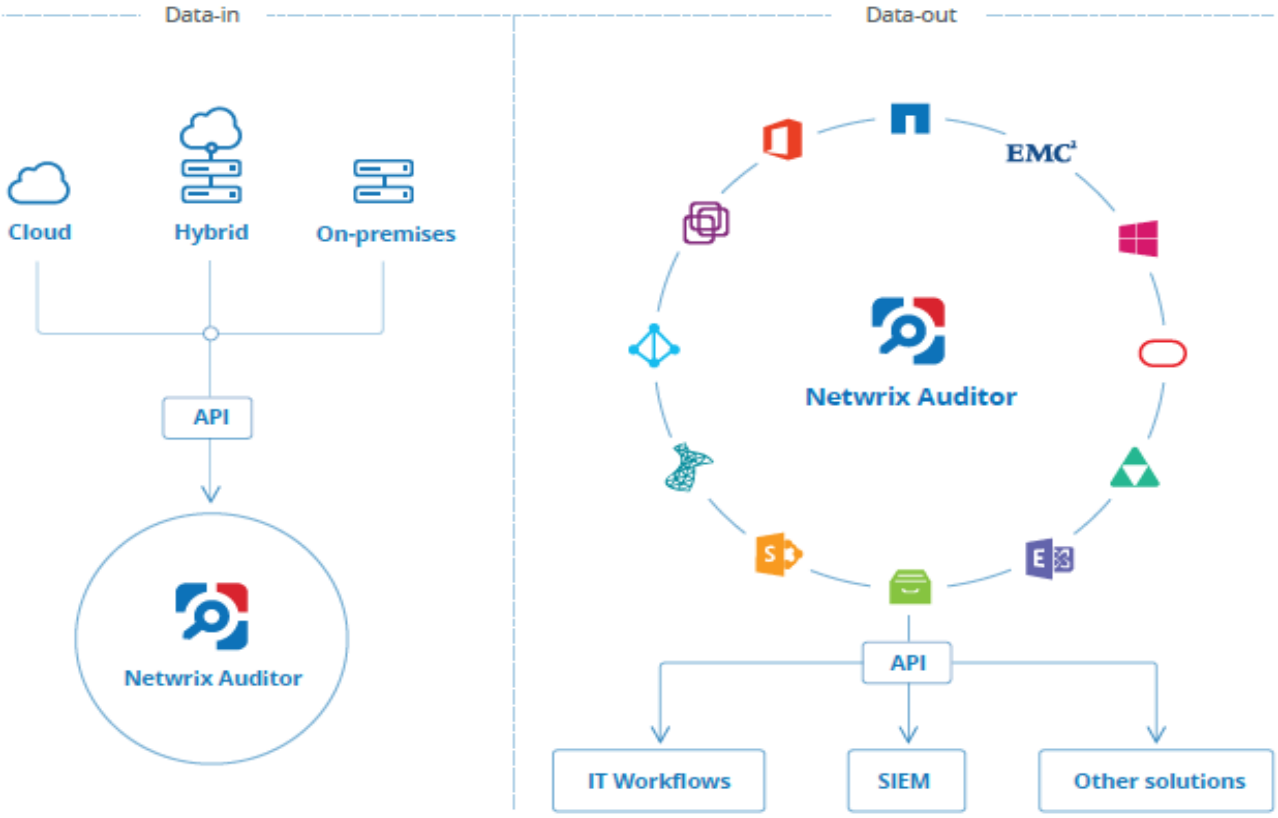
By default, data is written to both the Audit Database and the Long-Term Archive that is designed to store data in a compressed format for a longer period of time . With two-tiered AuditArchive you can store your data as long as required in the Long-Term Archive (by default, 120 months), but keep your operational storage fast and clean and use it for browsing recent data (by default, 180 days). At the same time, Netwrix Auditor allows you to extract data from the Long-Term Archive and import it to the Audit Database if you want to investigate past issues.

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**: Solidify security and meet regulatory compliance standards by enabling visibility into what is going on in any third-party application.
- **Data out**: Further automate your business processes, IT security and operations workflows by enriching third-party solutions with actionable audit data.



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 and access audit data from remote computers. Also, Netwrix prepares add-ons—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 Server 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 the Netwrix Auditor main page, navigate to **Integrations**.
2. Make sure the **Leverage Integration API** option is set to *"On"*.
3. Click **Modify** under the **API settings** section and specify a port number. 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, 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. Also, you can write data to any other database associated with a monitoring plan created in the Netwrix Auditor client.

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

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

See Netwrix Auditor Administration 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

5.1. Account Permissions

Netwrix Auditor restricts control to its configuration and data collected by the product. Role-based access system ensures that only relevant employees and services can access the exact amount of data they need. To be able to retrieve activity records or supply data to the Audit Database, an account must be assigned a role in the product. See Netwrix Auditor Administration Guide for more information about role delegation and assignment procedure.

To...	Required role
Retrieve all activity records and write data	The user must be assigned the **Global administrator** role in the product, or be a member of the **Netwrix Auditor Administrators** group on the computer that hosts Netwrix Auditor Server.
Retrieve all activity records	The user must be assigned the **Global reviewer** role in the product or be a member of the **Netwrix Auditor Client Users** group on the computer that hosts Netwrix Auditor Server.
Retrieve activity records within a limited scope	The user must be assigned the **Reviewer** role on a monitoring plan or folder with plans. In this case, Netwrix Auditor Server will retrieve only activity records the user is allowed to review according to the scope delegated (e.g., a scope can be limited to a single domain or file share).
Write activity records	The user must be assigned the **Contributor** role in the product.

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 2017 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 2017 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. Activity Records are retrieved according to the account's delegated scope. 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>PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <MonitoringPlan>
      <Name>AD Monitoring</Name>
      <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    </MonitoringPlan>
    <DataSource>Active Directory</DataSource>
```

```
    <Item>
      <Name>enterprise.local (Domain)</Name>
    </Item>
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
    <Action>Added</Action>
    <When>2017-02-14T15:42:34Z</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",
      "MonitoringPlan" : {
        "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
        "Name": "AD Monitoring"
        },
      "DataSource": "Active Directory",
      "Item": {"Name": "enterprise.local (Domain)"},
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
      "When": "2017-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
  ],
  "ContinuationMark": "PG5yPjxuIG49IntFNzA...PjwvYT48L24+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/">
  PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+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
```

```
"PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+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 interactive search available in the Netwrix Auditor client. See Netwrix Auditor Intelligence Guide for detailed instruction on how to search and filter audit data.

As the interactive 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. Activity Records are retrieved according to the account's delegated scope.

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 2017 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 2017 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, 2016 and March 16, 2017.

   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>
  <DataSource>Active Directory</DataSource>
  <Action>Added</Action>
  <ObjectType Operator="DoesNotContain">Group</ObjectType>
  <When>
   <From>2016-09-16T16:30:00+11:00</From>
   <To>2017-03-16T00:00:00Z</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",
    "DataSource": "Active Directory",
    "Action": "Added",
    "ObjectType": { "DoesNotContain": "Group"},
    "When": {
      "From": "2016-09-16T16:30:00+11:00",
      "To": "2017-03-16T00:00:00Z"
    }
  }
}
```

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. Activity Records are retrieved according to the account's delegated scope. 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>PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <MonitoringPlan>
      <Name>AD Monitoring</Name>
      <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    </MonitoringPlan>
    <DataSource>Active Directory</DataSource>
    <Item>
      <Name>enterprise.local (Domain)</Name>
    </Item>
```

```
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
    <Action>Added</Action>
    <When>2017-02-14T15:42:34Z</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",
      "MonitoringPlan" : {
        "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
        "Name": "AD Monitoring"
        },
      "DataSource": "Active Directory",
      "Item": {"Name": "enterprise.local (Domain)"},
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
      "When": "2017-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
  ],
  "ContinuationMark": "PG5yPjxuIG49IntFNzA...PjwvYT48L24+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>PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+PC9ucj4A</Continuatio
 nMark>
 <FilterList>
  <Who>Administrator</Who>
  <DataSource>Active Directory</DataSource>
  <Action>Added</Action>
  <ObjectType Operator="DoesNotContain">Group</ObjectType>
  <When>
   <From>2016-09-16T16:30:00+11:00</From>
   <To>2017-03-16T00:00:00Z</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": "PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+PC9ucj4A",
 "FilterList": {
  "Who": "Administrator",
  "DataSource": "Active Directory",
  "Action": "Added",
  "ObjectType": { "DoesNotContain": "Group"},
  "When": {
   "From": "2016-09-16T16:30:00+11:00",
   "To": "2017-03-16T00:00:00Z"
  }
 }
}
```

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 Audit Database and to the Long-Term Archive. By default, all imported data is written to a special **Netwrix_Auditor_API** database and recognized as the **Netwrix API** data source. This data is not associated with any monitoring plan in the product. You can associate Activity Records with a plan, in this case data will be written to a database linked to this plan. Make sure the plan you specify is already created in Netwrix Auditor, the **Netwrix API** data source is added to the plan and enabled for monitoring.

To feed data, send a POST request containing Activity Records. The user sending a request must be assigned the **Contributor** role in Netwrix Auditor. After feeding data to the Audit Database it will become available for search 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`
	`Server: Microsoft-HTTPAPI/2.0`
	`Content-Length: 0`
	`Content-Type: text/plain`
	`Date: Fri, 08 Apr 2017 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>
    <MonitoringPlan>
      <Name>Integrations and custom sources</Name>
    </MonitoringPlan>
    <Where>WKSWin12SQL</Where>
    <When>2017-02-19T03:43:49-11:00</When>
  </ActivityRecord>
  <ActivityRecord>
    <Action>Modified</Action>
    <ObjectType>Mailbox</ObjectType>
    <What>Shared Mailbox</What>
```

```
    <When>2017-02-10T14:46:00Z</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",
    "MonitoringPlan": {"Name": "Integrations and custom sources"},
    "What": "Databases\\ReportServer\\Stored Procedures\\dbo.sp_New",
    "Where": "WKSWin12SQL",
    "When": "2017-02-19T03:43:49-11:00"
  },
  {
    "Action": "Modified",
    "ObjectType": "Mailbox",
    "What": "Shared Mailbox",
    "When": "2017-02-10T14:46:00Z",
    "Where": "BLUPR05MB1940",
    "Who": "admin@enterprise.onmicrosoft.com",
    "DetailList": [
      {
        "PropertyName": "Custom_Attribute",
        "Before": "1",
        "After": "2"
      }
    ]
  }
]
```

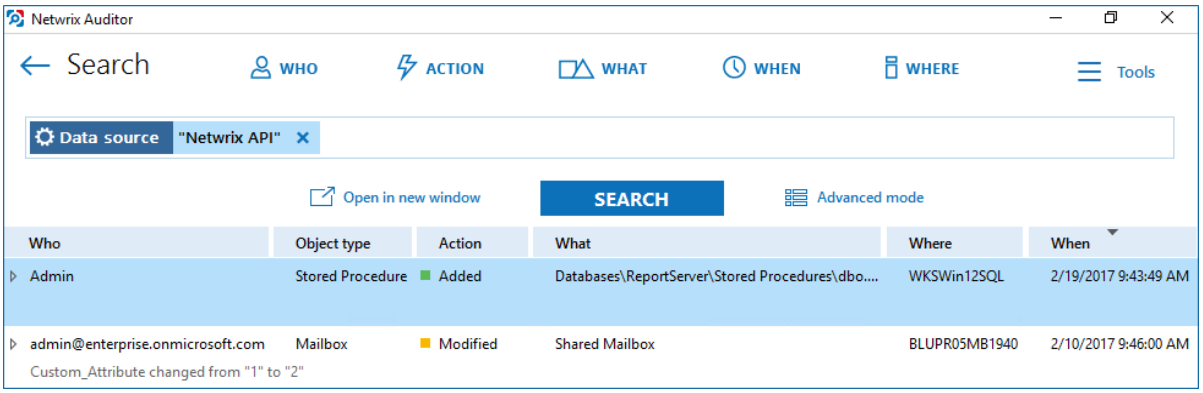
NOTE: Ensure to pass information about transferred data, including `Content-Type:application/xml` or `application/json` and encoding. The syntax greatly depends on the tool you use.

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

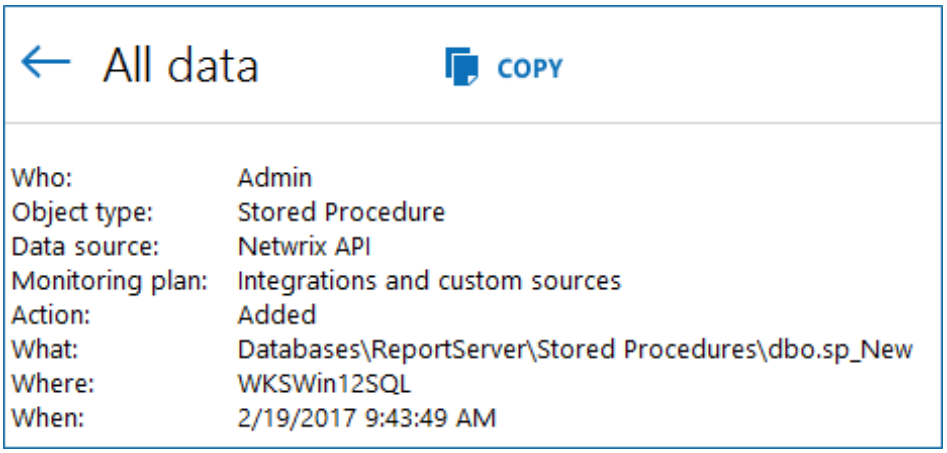
```
HTTP/1.1 200 OK
Server: Microsoft-HTTPAPI/2.0
Content-Length: 0
Content-Type: text/plain
Date: Fri, 08 Apr 2017 13:56:22 GMT
```

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 interactive search in the Netwrix Auditor client. For example:



NOTE: For input Activity Records, the data source in 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.

In Netwrix Auditor 9.0, Netwrix has updated API schemas. Make sure to check and update your custom scripts and add-ons. See Compatibility Notice for more information.

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 the computer where Netwrix Auditor Server resides, you can find XSD file under *Netwrix_Auditor_installation_folder\Audit Core\API Schemas*. 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/">
  PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+PC9ucj4A
</ContinuationMark>
```

Search Activity Records

```
<?xml version="1.0" standalone="yes"?>
<ActivityRecordSearch xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ContinuationMark>PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+PC9ucj4A</ContinuationMar
  k>
  <FilterList>
    <Who>Administrator</Who>
    <DataSource>Active Directory</DataSource>
    <Action>Added</Action>
    <ObjectType Operator="DoesNotContain">Group</ObjectType>
    <When>
      <From>2016-09-16T16:30:00+11:00</From>
      <To>2017-03-16T00:00:00Z</To>
    </When>
  </FilterList>
</ActivityRecordSearch>
```

JSON

Retrieve Activity Records

```
"PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A"
```

Search Activity Records

```
{
  "ContinuationMark": "PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A+PC9ucj4A",
  "FilterList": {
    "Who": "Administrator",
    "DataSource": "Active Directory",
    "Action": "Added",
    "ObjectType": { "DoesNotContain": "Group"},
    "When": {
      "From": "2016-09-16T16:30:00+11:00",
      "To": "2017-03-16T00:00:00Z"
    }
  }
}
```

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 *data source* 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 the computer where Netwrix Auditor Server resides, 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.
	
	`<?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="Label"/>`
	
	`<xs:simpleType name="ActionEnum">...</xs:simpleType>`
	
	`<xs:complexType name="StringFilter">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNTe">...</xs:complexType>`
	
	`<xs:complexType name="StringFilterNVa">...</xs:complexType>`
	
	`<xs:complexType name="ActionFilter">...</xs:complexType>`
	
	`<xs:complexType name="DateTimeFilter">...</xs:complexType>`
	
	`<xs:element name="ActivityRecordS">...</xs:element>`
	
	`</xs:schema>`

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.

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>
    <MonitoringPlan>My Hybrid Cloud enterprise</MonitoringPlan>
    <DataSource>Active Directory</DataSource>
    <DataSource Operator="StartsWith">Exchange</DataSource>
    <Action>Removed</Action>
    <Action>Added</Action>
    <ObjectType Operator="DoesNotContain">Group</ObjectType>
    <When>
      <From>2016-01-16T16:30:00+11:00</From>
      <To>2017-01-01T00:00:00Z</To>
    </When>
  </FilterList>
</ActivityRecordSearch>
```

JSON

```
{
  "FilterList": {
    "Who": { "NotEqualTo": "Administrator" },
    "MonitoringPlan": "My Hybrid Cloud enterprise",
    "DataSource": [ "Active Directory", { "StartsWith": "Exchange" } ],
    "Action": [ "Added", "Removed" ],
    "ObjectType": { "DoesNotContain": "Group" },
    "When": {
      "From": "2016-01-16T16:30:00+11:00",
      "To": "2017-01-01T00:00:00Z"
    }
  }
}
```

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>`
	`<DataSource Operator="NotEqualTo">Active Directory</DataSource>`
	`<What>User</What>`

JSON	`"Who" : { "Equals" : "Admin" },`
	`"DataSource" : { "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>`
`<DataSource>`
`SharePoint`
`</DataSource>`
`<Action Operator="NotEqualTo">`
`Read`
`</Action>`
• JSON:
	Retrieves all activity records where administrator made any actions on SharePoint, except Read.
	• XML:
	`<ActivityRecord>`
	`<Action>Added</Action>`
	`<MonitoringPlan>`
	`<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>`
	`<Name>Compliance</Name>`
	`</MonitoringPlan>`
	`<DataSource>SharePoint</DataSource>`

Filters	Matching Activity Records

"Who" : "Admin",
"DataSource" : "SharePoint",
"Action" : {
"NotEqualTo" : "Read"
}
	<Item>
	<Name>http://demolabsp:8080 (SharePoint farm)</Name>
	</Item>
	<ObjectType>List</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
	<What>http://demolabsp/lists/Taskslist</What>
	<When>2017-02-17T09:28:35Z</When>
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Removed</Action>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
	<Name>Compliance</Name>
	</MonitoringPlan>
	<DataSource>SharePoint</DataSource>
	<Item>
	<Name>http://demolabsp:8080 (SharePoint farm)</Name>
	</Item>
	<ObjectType>List</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D15857</RID>
	<What>http://demolabsp/lists/Old/Taskslist</What>
	<When>2017-02-17T09:28:35Z</When>
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	• JSON:
	{
	"Action": "Added",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "SharePoint",
	"Item": {"Name": "http://demolabsp:8080 (SharePoint farm)"},
	"ObjectType" : "List",
	"RID" : "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What" : "http://demolabsp/lists/Taskslist",
	"When" : "2017-02-17T09:28:35Z",
	"Where" : "http://demolabsp",
	"Who" : "Enterprise\\Administrator",
	"Workstation" : "172.28.15.126"
	},
	{

Filters	Matching Activity Records

	<pre>"Action" : "Removed",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "SharePoint",
	"Item": {"Name": "http://demolabsp:8080 (SharePoint farm)"},
	"ObjectType" : "List",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D15857",
	"What" : "http://demolabsp/lists/Old/Taskslist",
	"When" : "2017-02-17T09:28:35Z",
	"Where" : "http://demolabsp",
	"Who" : "Enterprise\\Administrator",
	"Workstation" : "172.28.15.126"
	}</pre>

• XML:
<pre><Who>Administrator</Who>
<Action>Added</Action></pre>• JSON:
<pre>"Who" : "Administrator",
"Action" : "Added"</pre>
	Retrieves all activity records where administrator added an object within any data source.
	• XML:
	<pre><ActivityRecord>
	<Action>Added</Action>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
	<Name>Compliance</Name>
	</MonitoringPlan>
	<DataSource>SharePoint</DataSource>
	<Item>
	<Name>http://demolabsp:8080 (SharePoint farm)</Name>
	</Item>
	<ObjectType>List</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
	<What>http://demolabsp/lists/Taskslist</What>
	<When>2017-02-17T09:28:35Z</When>
	<Where>http://demolabsp</Where>
	<Who>Enterprise\Administrator</Who>
	<Workstation>172.28.15.126</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Added</Action>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
	<Name>Compliance</Name>
	</MonitoringPlan>
	<DataSource>Exchange</DataSource>
	<Item>
	<Name>enterprise.local (Domain)</Name>
	</Item></pre>

Filters	Matching Activity Records

	<ObjectType>Mailbox</ObjectType>
	<RID>2016021116354759207E9DDCEEB674986AD30CD3D13F5DEA3</RID>
	<What>Shared Mailbox</What>
	<When>2017-02-10T14:46:00Z</When>
	<Where>eswks.enterprise.local</Where>
	<Who>Enterprise\Administrator</Who>
	</ActivityRecord>
	• JSON:
	{
	"Action" : "Added",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "SharePoint",
	"Item": {"Name": "http://demolabsp:8080 (SharePoint farm)"},
	"ObjectType": "List",
	"RID": "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What": "http://demolabsp/lists/Taskslist",
	"When": "2017-02-17T09:28:35Z",
	"Where": "http://demolabsp",
	"Who": "Enterprise\\Administrator",
	"Workstation": "172.28.15.126"
	},
	{
	"Action" : "Added",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource" : "Exchange",
	"Item": {"Name": "enterprise.local (Domain)"},
	"ObjectType" : "Mailbox",
	"RID": "2016021116354759207E9DDCEEB674986AD30CD3D13F5DEA3",
	"What": "Shared Mailbox",
	"When": "2017-02-10T14:46:00Z",
	"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 data source. • XML: <ActivityRecord> <Action>Added</Action> <MonitoringPlan>

Filters	Matching Activity Records

```
    <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    <Name>Compliance</Name>
  </MonitoringPlan>
  <DataSource>File Servers</DataSource>
  <Item>
    <Name>wks.enterprise.local (Computer)</Name>
  </Item>
  <ObjectType>Folder</ObjectType>
  <RID>2016021116354759207E9DDCEEB674986AD30CD3D13F5DDA3</RID>
  <What>Annual_Reports</What>
  <When>2017-02-10T14:46:00Z</When>
  <Where>wks.enterprise.local</Where>
  <Who>Enterprise\Admin</Who>
</ActivityRecord>
<ActivityRecord>
  <Action>Removed</Action>
  <MonitoringPlan>
    <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    <Name>Compliance</Name>
  </MonitoringPlan>
  <DataSource>Active Directory</DataSource>
  <Item>
    <Name>enterprise.local (Domain)</Name>
  </Item>
  <ObjectType>User</ObjectType>
  <RID>2016021116354759207E9DDCEEB674986AD30CD3D13F5DAA3</RID>
  <What>Anna.Smith</What>
  <When>2017-02-10T10:46:00Z</When>
  <Where>dc1.enterprise.local</Where>
  <Who>Enterprise\Analyst</Who>
  <Workstation>172.28.6.15</Workstation>
</ActivityRecord>
```

- JSON:

```
{
  "Action": "Added",
  "MonitoringPlan": {
    "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
    "Name": "Compliance"
  },
  "DataSource" : "File Servers",
  "Item": {"Name": "wks.enterprise.local (Computer)"},
  "ObjectType": "Folder",
  "RID": "2016021116354759207E9DDCEEB674986AD30CD3D13F5DDA3",
  "What": "Annual_Reports",
  "When": "2017-02-10T14:46:00Z",
  "Where": "wks.enterprise.local",
  "Who": "Enterprise\\Admin"
```

Filters	Matching Activity Records

	<pre>},
	{
	"Action": "Removed",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "Active Directory",
	"Item": {"Name": "enterprise.local (Domain)"},
	"ObjectType": "User",
	"RID": "20160211163547592O7E9DDCEEB674986AD30CD3D13F5DAA3",
	"What": "Anna.Smith",
	"When": "2017-02-10T10:46:00Z",
	"Where": "dc1.enterprise.local",
	"Who": "Enterprise\\Analyst",
	"Workstation": "172.28.6.15"
	}</pre>

• XML:
<pre><When>
<LastSevenDays/>
</When>
<When>
<From>
2017-01-16T16:30:00Z
</From>
<To>
2017-02-01T00:00:00Z
</To>
</When></pre>
• JSON:
<pre>"When" : [
"LastSevenDays",
{
"From" : "2017-01-16T16:30:00Z",
"To" : "2017-02-01T00:00:00Z"
}
]</pre>
	Retrieves all activity records for all data sources and users within a specified data range:
	• January 16, 2017 — February 1, 2017
	• March 11, 2017 — March 17, 2017 (assume, today is March, 17).
	• XML:
	<pre><ActivityRecord>
	<Action>Modified</Action>
	<MonitoringPlna>My Cloud</MonitoringPlan>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23701}</ID>
	<Name>My Cloud</Name>
	</MonitoringPlan>
	<DataSource>Exchange Online</DataSource>
	<Item>
	<Name>mail@corp.onmicrosoft.com (Office 365 tenant)</Name>
	</Item>
	<ObjectType>Mailbox</ObjectType>
	<RID>20160217093959797O997D56DDA034420B9044249CC15EC5A</RID>
	<What>Shared Mailbox</What>
	<When>2017-03-17T09:37:11Z</When>
	<Where>BLUPR05MB1940</Where>
	<Who>admin@corp.onmicrosoft.com</Who>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Successful Logon</Action></pre>

Filters	Matching Activity Records

```
  <MonitoringPlan>
    <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    <Name>Compliance</Name>
  </MonitoringPlan>
  <DataSource>Logon Activity</DataSource>
  <Item>
    <Name>enterprise.local (Domain)</Name>
  </Item>
  <ObjectType>Logon</ObjectType>
  <RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
  <What>stationexchange.enterprise.local</What>
  <When>2017-02-17T09:28:35Z</When>
  <Where>enterprisedc1.enterprise.local</Where>
  <Who>ENTERPRISE\Administrator</Who>
  <Workstation>stwin12R2.enterprise.local</Workstation>
</ActivityRecord>
```

- JSON:

```
{
  "Action" : "Modified",
  "MonitoringPlan" : "My Cloud",
  "MonitoringPlan": {
    "ID": "{42F64379-163E-4A43-A9C5-4514C5A23701}",
    "Name": "My Cloud"
  },
  "DataSource": "Exchange Online",
  "Item": {
    "Name": "mail@corp.onmicrosoft.com (Office 365 tenant)"
  },
  "ObjectType" : "Mailbox",
  "RID" : "20160217093959797097D56DDA034420B9044249CC15EC5A",
  "What" : "Shared Mailbox",
  "When" : "2017-03-17T09:37:11Z",
  "Where" : "BLUPR05MB1940",
  "Who" : "admin@corp.onmicrosoft.com"
},
{
  "Action" : "Successful Logon",
  "MonitoringPlan": {
    "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
    "Name": "Compliance"
  },
  "DataSource": "Logon Activity",
  "Item": {"Name": "enterprise.local (Domain)"},
  "ObjectType": "Logon",
  "RID" : "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
  "What" : "stationexchange.enterprise.local",
  "When" : "2017-02-17T09:28:35Z",
```

Filters	Matching Activity Records

	<pre>"Where" : "enterprisedc1.enterprise.local",
	"Who" : "ENTERPRISE\\Administrator",
	"Workstation" : "stwin12R2.enterprise.local"
	}</pre>

• XML:
<pre><DataSource>
Logon Activity
</DataSource></pre>• JSON:
<pre>"DataSource" : "Logon Activity"</pre>
	Retrieves all activity records for Logon Activity data source irrespective of who made logon attempt and when it was made.
	• XML:
	<pre><ActivityRecord>
	<Action>Successful Logon</Action>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
	<Name>Compliance</Name>
	</MonitoringPlan>
	<DataSource>Logon Activity</DataSource>
	<Item>
	<Name>enterprise.local (Domain)</Name>
	</Item>
	<ObjectType>Logon</ObjectType>
	<RID>20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7</RID>
	<What>stationexchange.enterprise.local</What>
	<When>2017-02-17T09:28:35Z</When>
	<Where>enterprisedc1.enterprise.local</Where>
	<Who>ENTERPRISE\Administrator</Who>
	<Workstation>stwin12R2.enterprise.local</Workstation>
	</ActivityRecord>
	<ActivityRecord>
	<Action>Successful Logon</Action>
	<MonitoringPlan>
	<ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
	<Name>Compliance</Name>
	</MonitoringPlan>
	<DataSource>Logon Activity</DataSource>
	<Item>
	<Name>enterprise.local (Domain)</Name>
	</Item>
	<ObjectType>Logon</ObjectType>
	<RID>20160217093959797097D56DDA034420B9044249CC15EC5A</RID>
	<What>stationwin12r2.enterprise.local</What>
	<When>2017-02-17T09:37:11Z</When>
	<Where>enterprisedc2.enterprise.local</Where>
	<Who>ENTERPRISE\Analyst</Who>
	<Workstation>stwin12R2.enterprise.local</Workstation>
	</ActivityRecord></pre>• JSON:

Filters	Matching Activity Records

	{
	"Action" : "Successful Logon",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "Logon Activity",
	"Item": {"Name": "enterprise.local (Domain)"},
	"ObjectType" : "Logon",
	"RID" : "20160217093959797091D091D2EAF4A89BF7A1CCC27D158A7",
	"What" : "stationexchange.enterprise.local",
	"When" : "2017-02-17T09:28:35Z",
	"Where" : "enterprisedc1.enterprise.local",
	"Who" : "ENTERPRISE\\Administrator",
	"Workstation" : "stwin12R2.enterprise.local"
	},
	{
	"Action" : "Successful Logon",
	"MonitoringPlan": {
	"ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
	"Name": "Compliance"
	},
	"DataSource": "Logon Activity",
	"Item": {"Name": "enterprise.local (Domain)"},
	"ObjectType" : "Logon",
	"RID" : "201602170939597970997D56DDA034420B9044249CC15EC5A",
	"What" : "stationwin12r2.enterprise.local",
	"When" : "2017-02-17T09:37:11Z",
	"Where" : "enterprisedc2.enterprise.local",
	"Who" : "ENTERPRISE\\Analyst",
	"Workstation" : "stwin12R2.enterprise.local"
	}

9.2.3.1. Filters

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

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

Filter	Description	Supported Operators
Who	Limits your search to a specific user who made the change (e.g., *Enterprise\Administrator*, *administrator@enterprise.onmicrosoft.com*). 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 • 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
DataSource	Limits your search to the selected data source only (e.g., *Active Directory*). Max length: 1073741822.	• Contains (default) • DoesNotContain • Equals • NotEqualTo • StartsWith • EndsWith
Monitoring Plan	Limits your search to a specific plan— Netwrix Auditor object that governs data collection. Max length: 255.	• Contains (default) • DoesNotContain

Filter	Description	Supported Operators
		• Equals • NotEqualTo • StartsWith • EndsWith
Item	Limits your search to a specific item— object of monitoring—and its type provided in brackets. The following item types are available: • AD container • Computer • Domain • EMC Isilon • EMC VNX/VNXe • Integration • IP range • NetApp • Office 365 tenant • Oracle Database instance • SharePoint farm • SQL Server instance • VMware ESX/ESXi/vCenter • Windows file share 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 information specific to your data source, 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

Filter	Description	Supported Operators
		• EndsWith
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 • Failed 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 • Successful Logon • Logoff • Sent • Session end	• Equals (default) • NotEqualTo
When	Limits your search to a specified time range. Netwrix Auditor 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*, *John@domain.com*.
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 *Johnny@domain.com*.

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/"> <ActivityRecord> <Who>Who</Who> <ObjectType>Object Type</ObjectType> <Action>Action</Action> <What>What</What> <When>When</When> <Where>Where</Where> <MonitoringPlan> <ID>Unique ID</ID> <Name>Name</Name> </MonitoringPlan> <DataSource>Data source</DataSource> <Item> <Name>Item name (Item type)</Name> </Item> <DetailList> <Detail> <Before>Before Value</Before> <After>After Value</After> <PropertyName>Property</PropertyName> <Message>Text</Message> </Detail> </DetailList> </ActivityRecord> <ActivityRecord>...</ActivityRecord> </ActivityRecordList>

JSON	[
	{
	"Action": "Action",
	"MonitoringPlan": {
	"ID": "Unique ID",
	"Name": "Name"
	},
	"DataSource": "Data source",
	"Item": {"Name": "Item name (Item type)"},
	"DetailList": [
	{
	"Before": "Before Value",
	"After": "After Value",
	"PropertyName": "Property",
	"Message": "Text"
	}
	],

Format	Example

```
    "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 the computer where Netwrix Auditor Server resides, 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 `MonitoringPlan` element contains sub-elements such as `Name` and `ID`, the `Item` element contains `Name`. Both `MonitoringPlan` and `Item` are optional for input Activity Records. The `DetailList` element is optional too, 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 data source is automatically set to **Netwrix API**. NOTE: **minOccurs="0"** indicates that element is optional and may be absent when writing data to the Audit Database.
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 data source is automatically set to **Netwrix API**.

9.3.2. Example

The examples below show an output Activity Record.

XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<ActivityRecordList xmlns="http://schemas.netwrix.com/api/v1/activity_records/">
  <ActivityRecord>
    <Action>Modified</Action>
    <MonitoringPlan>
      <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
      <Name>Compliance</Name>
    </MonitoringPlan>
    <DataSource>Exchange Online</DataSource>
    <Item>
      <Name>mail@enterprise.onmicrosoft.com (Office 365 tenant)</Name>
    </Item>
    <ObjectType>Mailbox</ObjectType>
    <What>Shared Mailbox</What>
    <When>2017-03-17T09:37:11Z</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",
    "MonitoringPlan": {
      "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
      "Name": "Compliance"
    },
    "DataSource": "Exchange Online",
    "Item": {"Name": "mail@enterprise.onmicrosoft.com (Office 365 tenant)"},
    "ObjectType": "Mailbox",
    "What": "Shared Mailbox",
    "When": "2017-03-17T09:37:11Z",
    "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*, *Admin@enterprise.onmicrosoft.com*).
Action	Yes	—	Activity captured by Netwrix Auditor (varies depending on the data source): • Added • Removed • Modified • Read • Moved • Renamed • Checked in • Discard check out • Failed 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 • Successful Logon • Logoff • Sent • Session end
What	Yes	nvarchar	A specific object that was changed (e.g., *NewPolicy*).

Element	Mandatory	Datatype	Description
		max	
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)
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*).
Monitoring Plan	No	nvarchar 255	The Netwrix Auditor object that responsible for monitoring of a given data source and item. Sub-elements: Name and ID. **NOTE:** If you provide a monitoring plan name for input Activity Records, make sure the plan is created in Netwrix Auditor, the **Netwrix API** data source is added to the plan and enabled for monitoring. In this case, data will be written to the database associated with this plan.
DataSource	No	nvarchar max	IT infrastructure monitored with Netwrix Auditor (e.g., *Active Directory*). For input Activity Records, the data source is automatically set to **Netwrix API**.
Item	No	nvarchar max	The exact object that is monitored (e.g., a domain name, SharePoint farm name) or integration name. Sub-element: Name. The item type is added inside the name value in brackets

Element	Mandatory	Datatype	Description
			(e.g., *enterprise.local (Domain)*). For input Activity Records, the type is automatically set to **Integration**, you do not need to provide it. The output Activity Records may contain the following item types depending on the monitoring plan configuration: • AD container • Computer • Domain • EMC Isilon • EMC VNX/VNXe • Integration • IP range • NetApp • Office 365 tenant • Oracle Database instance • SharePoint farm • SQL Server instance • VMware ESX/ESXi/vCenter • Windows file share **NOTE:** If you provide an item name for input Activity Records, make sure this item is included in the monitoring plan within the **Netwrix API** data source. If you specify an item that does not exist, data will be written to the plan's database anyway but will not be available for search using the **Item** filter.
Workstation	No	nvarchar max	An originating workstation from which the change was made (e.g., *WKSwin12.enterprise.local*).
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.
DetailList	No	—	Information specific to the data source, e.g., assigned permissions, before and after values, start and end dates. References details.
Detail sub-elements (provided that DetailList exists)			
PropertyName	Yes	nvarchar 255	The name of a modified property.
Message	No	string	Object-specific details about the change. Message is included in output Activity Records only.
Before	No	ntext	The previous value of the modified property.

Element	Mandatory	Datatype	Description
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 license has expired. • Internal error occurred.	Error retrieving Activity Records from the Audit Database: • Netwrix Auditorlicense has expired. • The **Netwrix Auditor Archive Service** is unreachable. Try restarting the service on the computer that hosts Netwrix Auditor Server. • Internal error occurred.

Code	Status	Write Activity Records	Retrieve, search Activity Records
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>`

JSON	`{`
	`  "ErrorList": [`
	`  {`
	`  "Category": "Category",`

Format	Example

```
    "Description": "Error Description",
    "Location": "Error Location"
   }
  ]
}
```

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

Request	Error details returned

Invalid request:

XML:

```
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
```

```
<?xml version="1.0" encoding="utf-8"?>
<ActivityRecordSearch xmlns="http://schemas.
netwrix.com/api/v1/activity_records/">
  <FilterList>
    <Who>Administrator</Who>
    <DataSource>Active Directory
    <Action>Modified</Action>
  </FilterList>
</ActivityRecordSearch>
```

- JSON:

```
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.json
```

```
{
  "FilterList": {
    "Who": "Administrator",
    "DataSource": "Active Directory
    "Action": "Added"
  }
}
```

400 Bad Request

- XML:

```
<?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 'DataSource'
    </Description>
  </Error>
</ErrorList>
```

- JSON:

NOTE: If JSON is corrupted, server returns 500 Internal Server Error with empty body.

Request	Error details returned

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

Valid request, but the Audit Database is unreachable: • XML: `curl https://WKSWin12R2:9699/ netwrix/api/v1/activity_ records/enum -u Enterprise\ NetwrixUser:NetwrixIsCool` • JSON: `curl https://WKSWin12R2:9699/ netwrix/api/v1/activity_ records/enum?format=json -u Enterprise\NetwrixUser: NetwrixIsCool`	500 Internal Server Error • XML: `<?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>` • JSON: `{ "ErrorList": [{ "Category": "ServerError",`

Request	Error details returned


```
    "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 contains 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.

Benefits:

- Centralize auditing and reporting of your IT environment—Netwrix Auditor unifies auditing of all IT systems across your on-premises, cloud or hybrid environment, and enables centralized reporting for security and compliance.
- Get the most from your SIEM investment—To maximize SIEM value, Netwrix Auditor increases the signal-to-noise ratio and feeds your HP ArcSight, Splunk, IBM QRadar or any other SIEM solution with much more granular audit data.
- Automate your IT workflows—Automate and improve your change management, service desk and other critical IT workflows by feeding them audit data from Netwrix Auditor.

Review the following for additional information:

- Available Add-Ons
- Use Add-Ons

11.1. Available Add-Ons

At the time of Netwrix Auditor 9.5 release, the following add-ons were verified and posted in Add-ons Store.

NOTE: In Netwrix Auditor 9.0, Netwrix has updated API schemas and older add-ons may become inoperable in 9.5. If you use add-ons that were released at the time of 8.0 or 8.5, make sure to download the latest add-on version in the Add-on Store. See Compatibility Notice for more information.

Name	Technology	Data in/out	Description
Add-on for Amazon Web Services	PowerShell	In	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.
CEF Export Add-on	PowerShell	Out	Exports Activity Records from the Audit

Name	Technology	Data in/out	Description
			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.
Event Log Export Add-on	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log—Netwrix_Auditor_Integration. Use this script to integrate data collected by Netwrix Auditor with SIEM solutions that use events as input data.
Add-on for ArcSight	PowerShell	Out	Exports Activity Records from the Audit Database to ArcSight in its native CEF format. Use this script to integrate Netwrix Auditor with ArcSight and extend auditing possibilities.
Add-on for RADIUS server	PowerShell	In	Exports RADIUS logon events from the Security event log and feeds them to the Audit Database. Use this script to track logon activity on servers with RADIUS protocol enabled. The add-on works in collaboration with Netwrix Auditor for Active Directory, collecting additional data that augments the data collected by Netwrix Auditor. Aggregating data into a single audit trail simplifies logon activity analysis and helps you keep tabs on your IT infrastructure.
Add-on for Splunk	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with Splunk and extend auditing possibilities.
Add-on for IBM QRadar	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with IBM QRadar and extend auditing possibilities.
Add-on for AlienVault USM	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with AlienVault USM and extend auditing

Name	Technology	Data in/out	Description
			possibilities.
Add-on for Solarwinds Log & Event Manager	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with Solarwinds Log & Event Manager and extend auditing possibilities.
Add-on for Intel Security	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with Intel Security and extend auditing possibilities.
Add-on for LogRhythm	PowerShell	Out	Exports Activity Records from the Audit Database to a custom Windows event log. Use this script to integrate Netwrix Auditor with LogRhythm and extend auditing possibilities.
Add-on for Cisco Network Devices	C#	In	Implemented as a service, the add-on listens to UDP port and feeds events from Cisco network devices to the Audit Database. The add-on comes with processing rules for Cisco ASA and IOS devices. Use this add-on if you want to include Cisco activity in your audit trail.
Add-on for Generic Linux Syslog	C#	In	Implemented as a service, the add-on listens to UDP port and feeds events from Syslog-based devices to the Audit Database. The add-on comes with processing rules for rsyslog messages. Use this add-on if you want to include Red Hat Enterprise Linux 7 and 6, SUSE Linux Enterprise Server 12, openSUSE 42, and Ubuntu 16, etc., activity in your audit trail.
Add-on for Privileged User Monitoring on Linux and Unix	C#	In	Implemented as a service, the add-on listens to UDP port and feeds events from Syslog-based devices to the Audit Database. The add-on comes with processing rules for rsyslog messages. Use this add-on if you want to detect SUDO commands and remote access (SSH) on Red Hat Enterprise Linux 7 and 6, SUSE Linux Enterprise Server 12, openSUSE 42, and Ubuntu 16, etc.

Name	Technology	Data in/out	Description
Add-on for ServiceNow Incident Management	C#	Out	Implemented as a service, the add-on facilitates data transition from Netwrix Auditor and automates ticket creation in ServiceNow Istanbul and Helsinki.

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.

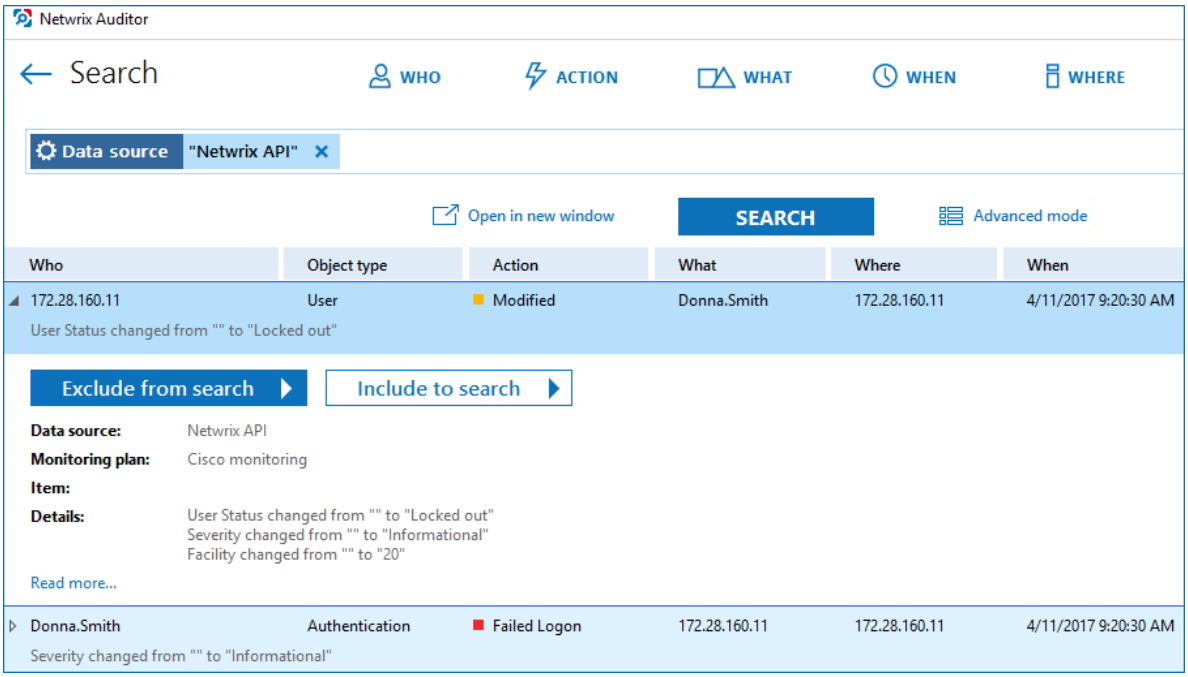
Refer to Security for detailed instructions on how to assign a new certificate and enable trust on remote computers.

11.2. Use Add-Ons

To use the add-on

1. Check prerequisites. Since the add-ons work only in combination with Netwrix Auditor, make sure the product and its Audit Database are configured, roles are assigned in the product. Some add-ons may require additional components to be installed in your system or options configured.
2. Define parameters. Before running or scheduling the add-on, you must define connection details: Netwrix Auditor Server host, user credentials, etc. You can skip or define parameters depending on your add-on, execution scenario, and security policies.
3. Choose appropriate execution scenario. Select where and who is going to execute the add-on.
4. Run the PowerShell-based add-on from a command line. Start **Windows PowerShell** and provide parameters. First provide a path to your add-on followed by script parameters with their values. Each parameter is preceded with a dash; a space separates a parameter name from its value. You can skip some parameters—the script uses a default value unless a parameter is explicitly defined.

   For add-ons implemented as a service, run the installation file that will deploy and start the service.
5. Review the results. For add-ons that import data to the Audit Database, search Activity Records in the Netwrix Auditor client. For example:



6. For PowerShell-based add-ons, schedule a daily task to ensure your audit data is always up-to-date.

Netwrix creates quick-start guides to help you incorporate add-ons in your daily routine. Each guide contains detailed instructions for running the add-on.

12. IIS Forwarding

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

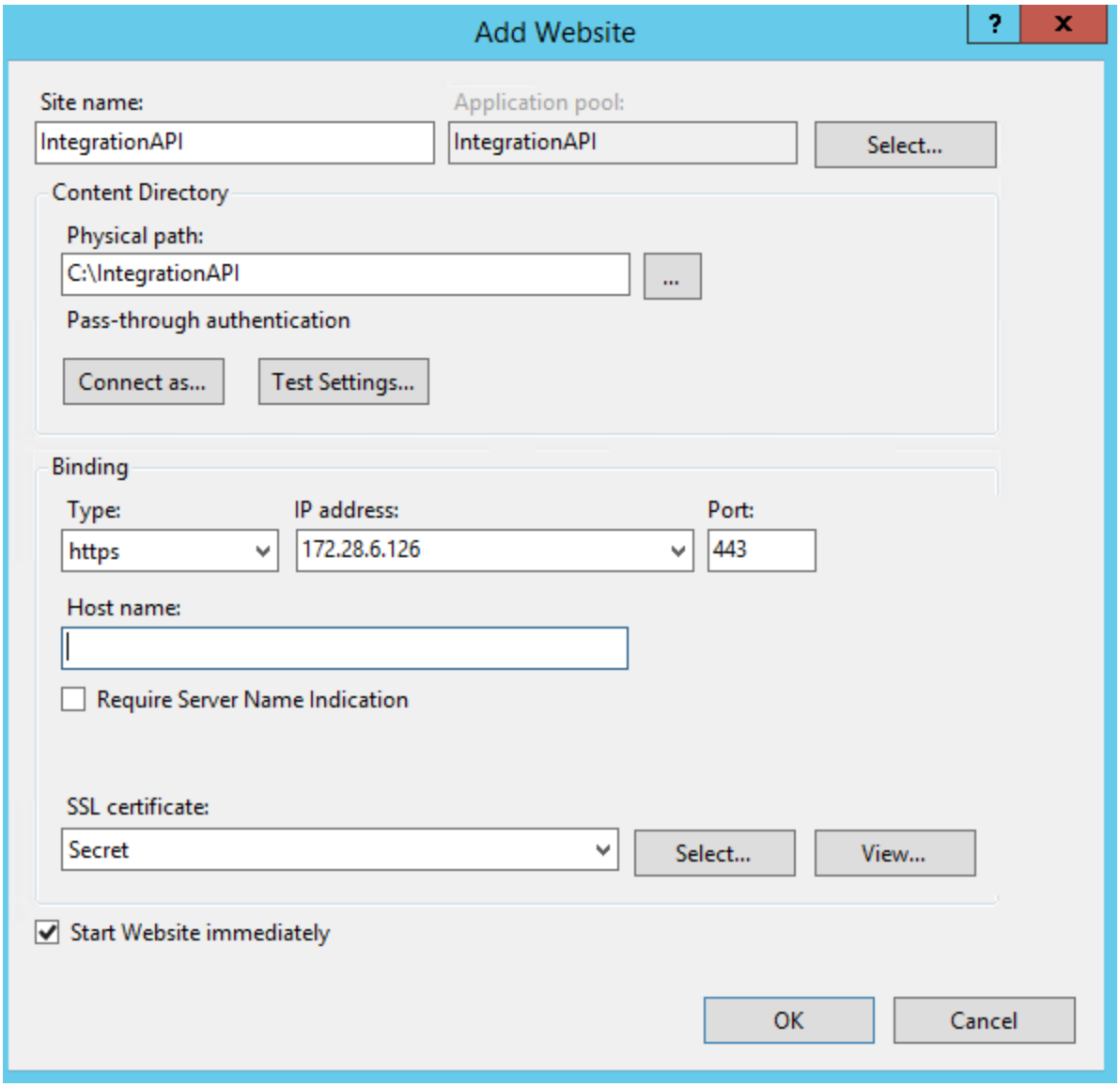
You can create a website in IIS and use it as a proxy for forwarding API requests. This is handy if for security reasons you do not want to make the Netwrix Auditor Server host name or address public. In this case, you can create a website with a short and user-friendly name and configure it to redirect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

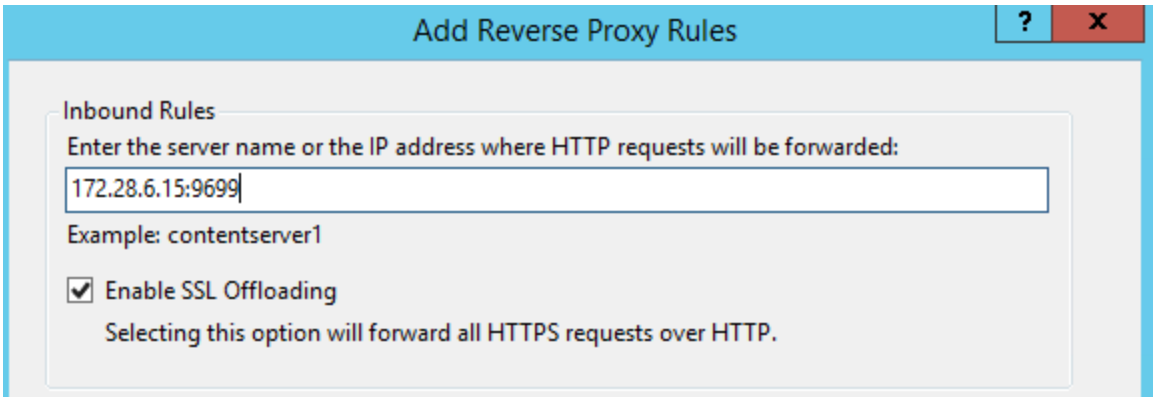
12.1. Configure IIS Forwarding

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

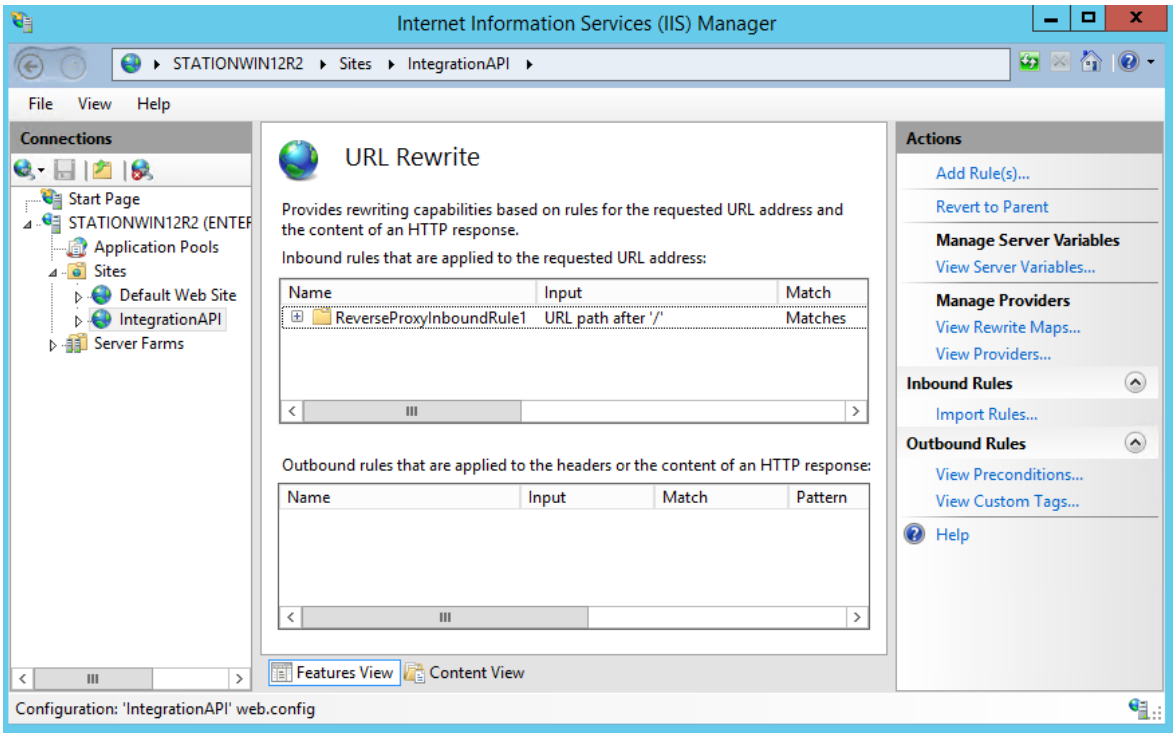
1. Make sure the **Web Server** role is installed on your server. Install the following components:
   - Application Request Routing
   - URL Rewrite
2. Create IIS website. To do this, navigate to **Start → Windows Administrative Tools** (Windows Server 2016) or **Administrative Tools** (Windows 2012 R2 and below) → **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>PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A</ContinuationMark>
  <ActivityRecord>
    <MonitoringPlan>
      <Name>AD Monitoring</Name>
      <ID>{42F64379-163E-4A43-A9C5-4514C5A23798}</ID>
    </MonitoringPlan>
    <DataSource>Active Directory</DataSource>
    <Item>
      <Name>enterprise.local (Domain)</Name>
    </Item>
    <ObjectType>user</ObjectType>
    <RID>20160215110503420B9451771F5964A9EAC0A5F35307EA155</RID>
    <What>\local\enterprise\Users\Jason Smith</What>
    <Action>Added</Action>
    <When>2017-02-14T15:42:34Z</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",
      "MonitoringPlan" : {
        "ID": "{42F64379-163E-4A43-A9C5-4514C5A23798}",
        "Name": "AD Monitoring"
        },
      "DataSource": "Active Directory",
      "Item": {"Name": "enterprise.local (Domain)"},
      "ObjectType": "user",
      "RID": "20160215110503420B9451771F5964A9EAC0A5F35307EA155",
      "What": "\\local\\enterprise\\Users\\Jason Smith",
```

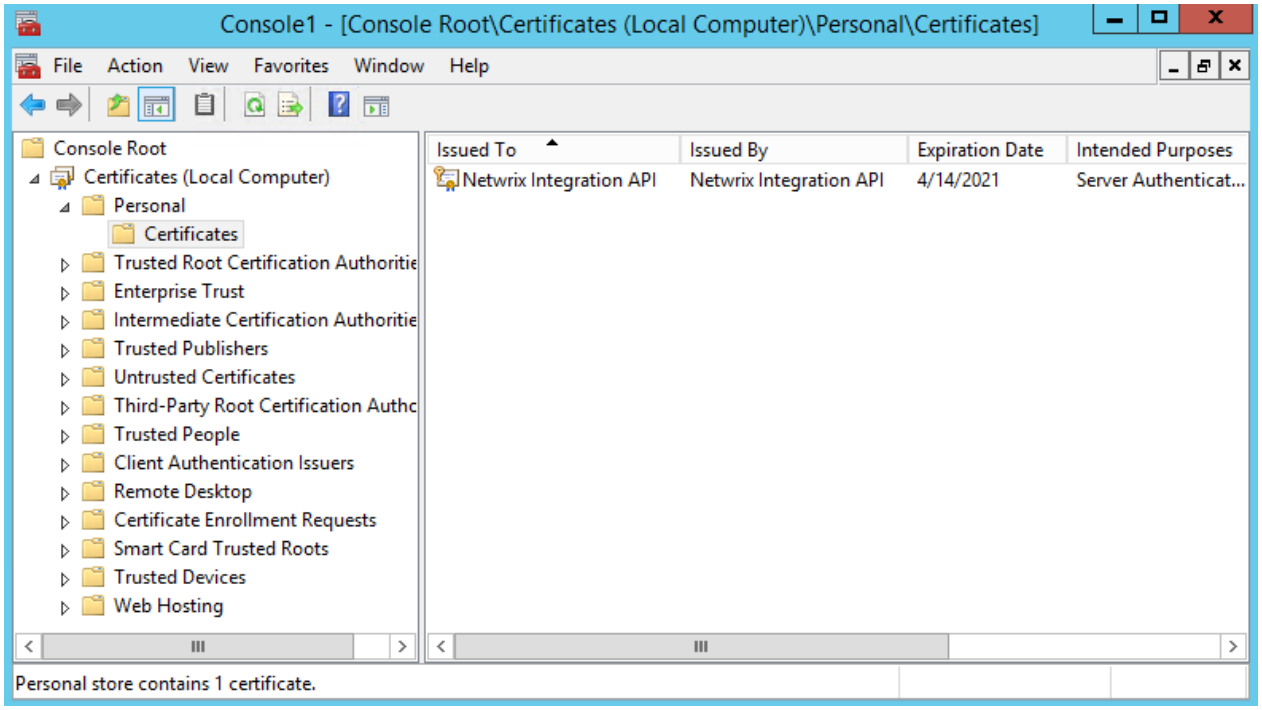
```
      "When": "2017-02-14T15:42:34Z",
      "Where": "EnterpriseDC1.enterprise.local",
      "Who": "ENTERPRISE\\Administrator",
      "Workstation": "EnterpriseDC1.enterprise.local"
    },
    {...},
    {...}
  ],
  "ContinuationMark": "PG5yPjxuIG49IntFNzA...PjwvYT48L24+PC9ucj4A"
}
```


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 Server resides, 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. 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 **Integrations** page in Netwrix Auditor.
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.
	  • validFrom—Optional, defines a certificate start date. By default, today.
	  • validTo—Optional, defines a certificate expiration date. By default, 5

To...	Execute...

	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`

14. Compatibility Notice

In Netwrix Auditor 9.0, Netwrix has updated API schemas. The scripts and add-ons designed for Netwrix Auditor 8.0 – 8.5 might become inoperable in Netwrix Auditor 9.5, while new add-ons designed for 9.0 and 9.5 cannot run at Netwrix Auditor 8.0 – 8.5.

Make sure to check your product version, and then review and update your add-ons and scripts leveraging Netwrix Auditor Integration API. Download the latest add-on version in the Add-on Store.

Property in 8.0 – 8.5	New property in 9.0 and above

• XML:
`<AuditedSystem></AuditedSystem>`
• JSON:
`"AuditedSystem"`
	• XML:
	`<DataSource></DataSource>`
	• JSON:
	`"DataSource"`

• XML:
`<ManagedObject></ManagedObject>`
• JSON:
`"ManagedObject"`
	• XML:
	`<MonitoringPlan>`
	`<Name>Name</Name>`
	`<ID>Unique ID</ID>`
	`</MonitoringPlan>`
	• JSON:
	`"MonitoringPlan" : {`
	`"ID": "{Unique ID}",`
	`"Name": "Name"`
	`}`
	NOTE: Now the MonitoringPlan contains two sub-entries: ID and Name. The ID property is optional and is assigned automatically by the product.

—	• XML:
	`<Item>`
	`<Name>Item name</Name>`
	`</Item>`
	• JSON:
	`"Item": {"Name": "Item name"}`

To learn more about input and output Activity Record structure, refer to Activity Records.

Index