
Micro Focus Security ArcSight Logger

Software Version: 7.3

Web Services API Guide



Legal Notices

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK

<https://www.microfocus.com>

Copyright Notice

Confidential computer software. Valid license from Micro Focus required for possession, use or copying. The information contained herein is subject to change without notice.

The only warranties for Micro Focus products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein.

No portion of this product's documentation may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's internal use, without the express written permission of Micro Focus.

Notwithstanding anything to the contrary in your license agreement for Micro Focus ArcSight software, you may reverse engineer and modify certain open source components of the software in accordance with the license terms for those particular components. See below for the applicable terms.

U.S. Governmental Rights. For purposes of your license to Micro Focus ArcSight software, "commercial computer software" is defined at FAR 2.101. If acquired by or on behalf of a civilian agency, the U.S. Government acquires this commercial computer software and/or commercial computer software documentation and other technical data subject to the terms of the Agreement as specified in 48 C.F.R. 12.212 (Computer Software) and 12.211 (Technical Data) of the Federal Acquisition Regulation ("FAR") and its successors. If acquired by or on behalf of any agency within the Department of Defense ("DOD"), the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of the Agreement as specified in 48 C.F.R. 227.7202-3 of the DOD FAR Supplement ("DFARS") and its successors. This U.S. Government Rights Section 18.11 is in lieu of, and supersedes, any other FAR, DFARS, or other clause or provision that addresses government rights in computer software or technical data.

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number
- Document Release Date, which changes each time the document is updated
- Software Release Date, which indicates the release date of this version of the software

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<https://www.microfocus.com/support-and-services/documentation>

Support

Contact Information

Phone	A list of phone numbers is available on the Technical Support Page: https://softwaresupport.softwaregrp.com/support-contact-information
Support Web Site	https://www.microfocus.com/en-us/support
ArcSight Product Documentation	https://www.microfocus.com/documentation/arcSight/

About this PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online help.

Contents

- Overview 8

- Chapter 1: Logger’s SOAP Web Services 10
 - Setting Up Your Development Environment 10
 - Accessing the SOAP Web Services 10
 - Setting an Authentication Token 11
 - Using Curl Method to Authenticate 11
 - Uploading CSR using Curl Method 13

- Chapter 2: SOAP Login Service 17
 - extendSession 17
 - getVersion 17
 - logout 17
 - login 17

- Chapter 3: SOAP Search Service 19
 - How the Search API Works 20
 - Returning Specific Fields in Search Results 20
 - endSearch 21
 - getDataforRowlds 21
 - getHeader 22
 - getNextTuples 22
 - hasMoreTuples 23
 - startSearch 23
 - Example: Searching for Events 24

- Chapter 4: SOAP Report Service 27
 - getDeviceGroups 27
 - getDevices 27
 - getDevicesInDeviceGroup 27
 - getReportGroups 27
 - getSubGroups 28
 - getReportsInGroup 28

getStorageGroups	28
runReport	28
Example: Running a Report	30
Example: Passing Parameters when Running a Report	34
Chapter 5: Logger's RESTful Web Services	37
About Logger's RESTful Web Services	37
Error Messages	37
Example Search API Error Messages	37
Supported REST Web Service Clients	38
Example: Making a Login Request Using Perl Script	38
Example: Making a Login request using Curl	39
Chapter 6: RESTful Login Service	40
login	40
Resource URL	40
Parameters	40
Response	41
Example Login Request	41
logout	41
Resource URL	41
Parameters	42
Response	42
Example Logout Request	42
Chapter 7: RESTful Search Service	43
http status codes	43
date/time format	43
search	44
Resource URL	44
Parameters	45
Response	47
Status	47
Resource URL	47
Parameters	47

Response	48
Example	48
histogram	49
Resource URL	49
Parameters	49
Response	49
Example	50
drilldown	50
Resource URL	50
Parameters	50
Response	51
Example	51
events	51
Resource URL	51
Parameters	52
Response	52
Example	53
raw_events	53
Resource URL	53
Parameters	54
Response	54
Example	54
chart_data	55
Resource URL	55
Parameters	55
Response	56
Example 1	56
stop	56
Resource URL	57
Parameters	57
Response	57
Example	57
close	57
Resource URL	58
Parameters	58
Response	58
Example	58

Example: Running a RESTful Search	58
Example: Returning Aggregate Search Data	61
Publication Status	69
Send Documentation Feedback	70

Overview

This document describes the Web services that come included with your installation of Micro Focus ArcSight Logger, version 7.3.



IMPORTANT: You can verify the type of Logger you have from the console, by executing either or both of these commands (as needed):

```
cat /etc/arcsight_model
```

```
cat /etc/OpenText_model
```

The output of these commands would be either your appliance model (**L7700** or **L8000**) or **No such file or directory** if you have a Software Logger.



Note: For simplicity, all types of Logger are called *Logger* in this document, except where noted.

The Logger Service Layer exposes Logger functions as Web services. By consuming the exposed Web services, you can integrate Logger functionality in your own applications. For example, you will be able to create programs that execute searches on stored Logger events or run Logger reports, and feed them back to your third-party system. Logger supports both SOAP-based and REST-based Web services.

The Service Layer uses a SOA (Service Oriented Architecture) or ROA (REST Oriented Architecture) that supports multiple Web service clients written in different languages. The ROA also supports standard REST clients.



Note: The examples provided in this guide are for illustration only and may not work as-is in your environment. To learn more about writing a Web service client, refer to the documentation of the language you intend to use to write the client.

This guide provides information on the following topics:

- "[Logger's SOAP Web Services](#)" on page 10 provides information applicable to all of Logger's SOAP-based Web services.
- "[SOAP Login Service](#)" on page 17 provides information about Logger's SOAP-based Login Service, which enables you to log in to a Logger and establish an authentication token that is used for all search and report service calls.
- "[SOAP Search Service](#)" on page 19 provides information about Logger's SOAP-based Search Service, which enables you to run search queries on Logger.
- "[SOAP Report Service](#)" on page 27 provides information about Logger's SOAP-based Report Service, which enables you to run report on Logger.
- "[Logger's RESTful Web Services](#)" on page 37 provides information applicable to all of Logger's RESTful Web services.

- ["RESTful Login Service" on page 40](#) provides information about Logger's RESTful Login Service, which enables you to log in to a Logger and establish a session ID that is used for all requests.
- ["RESTful Search Service" on page 43](#) provides information about Logger's RESTful Search Service, which enables you to run search queries on Logger.

For general information on Logger searching and reporting, refer to the [ArcSight Logger 7.3 Administrator's Guide](#).

Chapter 1: Logger's SOAP Web Services

This section provides information that applies to all of Logger's SOAP-based Web services. It covers the following topics:

Setting Up Your Development Environment

Take into consideration the following requirements when setting up your development environment:

- Java Web service clients require Java 8
- All exposed Logger SOAP Web services are TLS/SSL-secured, therefore, import the Logger's certificate into your development/runtime environment. The certificate option could be a temporary certificate authority (CA), a self-signed certificate, or a signed certificate from a trusted CA. Ask your ArcSight administrator which certificate option was chosen during installation and import that certificate into your development JRE's `jre/lib/security/cacerts`.

- Include these jar files in your Java classpath: `coma-infrastructure.jar` and `core-ws-client.jar`. The .jar files are located in the installation directory of the installation bits. Follow the paths displayed below:

For Software Logger or Logger Appliance L8000:

```
logger_install_path>/current/arcsight/aps/jarball/jars/coma-infrastructure-1.5.2.release.xx.xx.jar
```

```
logger_install_path>/current/arcsight/aps/jarball/jars/core-ws-client-1.6.0.release.xx.xx.jar
```

For Logger Appliance L7700:

```
logger_install_path>/aps/webapps/storage-service/WEB-INF/lib/coma-infrastructure-1.5.2.release.xx.jar
```

```
logger_install_path>/cli/lib/core-ws-client-1.6.0.release.xx.jar
```

Accessing the SOAP Web Services

The Service Layer's Web Service Description Language (WSDL) files are XML-formatted documents describing Logger services, one WSDL file for each service. WSDLs are used to generate clients automatically. Programmers who are writing their own stubs instead of using the SDK can refer to the WSDLs to get information about Logger services.

To access the SOAP Web services:

1. Install Logger 7.3.
2. Write a Web services client using a language of your choice, such as Java, Perl, or Python.

Use the following WSDL to access Logger's SOAP Web services:

On a Logger Appliance L7700:

```
https://<LoggerHost or  
IPAddress>/soap/services/<ServiceName>/<ServiceName>.wsdl
```

On a Software Logger or a Logger Appliance L8000:

```
https://<LoggerHost or IPAddress>:<port_  
number>/soap/services/<ServiceName>/<ServiceName>.wsdl
```

Where:

- **<LoggerHost or IPAddress>** is the hostname or IP address of the Logger.
- **<port_number>** is the port that you specify in the URL when connecting to a software Logger.
- **<ServiceName>** is the name of Web service you want to access.

For more information about LoginService, see ["SOAP Login Service" on page 17](#). For more information about ReportService, see ["SOAP Search Service" on page 19](#). For more information about SearchService, see ["SOAP Report Service" on page 27](#).

Setting an Authentication Token

All API calls require you to enter an authentication token that identifies the Logger session where the call will run. You set the authentication token when you log into a Logger using the LoginService login call.

For example, you can set the authentication token in this way:

```
authToken = LoginService.login("username", "password", 3600);
```

Using Curl Method to Authenticate

This topic applies to both Software Logger and the Logger Appliance.

This authentication method allows users to connect to API using Curl Command.



Note: Before starting with this process, configure the Client Certificate Authentication Method. To view the authentication commands, see the [ArcSight Logger 7.3 Administrator's Guide](#).



Tip: The client computer should have the "root CA" in its certificate repository; in case it is not configured use the "-k" parameter in the curl command.

The session id is required for all API requests and will be eventually used for the next subsequent commands. The "client certificates" are only required for the logging step.

Curl command and Logger with Client Certificate AND Local Password configuration in Login RESTful API

To connect using Curl, use the following parameters:

```
curl -H 'Accept: application/json' -X POST '[https://<IP or HOST>/core-service/rest/LoginService/login]' -k --key ./client2015key.pem --cert ./client2015.pem
```

If the certificate files are in the current directory where the command is executed, use "./" and the file name; otherwise, the command would fail. For example:

```
curl -H 'Accept: application/json' -X POST -d 'login=[usersample & password]=arcsight' '[https://<IP or HOST>/core-service/rest/LoginService/login]' -k --key ./client2015key.pem --cert ./client2015.pem
```

```
curl -k [https://<IP or HOST>/server/search] -H "Content-Type: application/json ; charset=[timezone]" -d '{ "search_session_id" : [number], "user_session_id" : "useTheSessionIdOfThePreviousCommand", "query" : "deviceVendor = ArcSight", "start_time" : "[yy-mm-ddThh:minutes:seconds.000-end time]", "end_time" : "[yy-mm-ddThh:minutes:seconds.000-end time]", "field_summary":true }'
```

Curl command and Logger with Client Certificate configuration in Login RESTful API

Unlike the previous method, "user" and "password" are not required. For example:

```
curl -H 'Accept: application/json' -X POST '[https://<IP or HOST>/core-service/rest/LoginService/login]' -k --key ./client2015key.pem --cert ./client2015.pem
```

```
curl -k [https://<IP or HOST>/server/search] -H "Content-Type: application/json ; charset=[timezone]" -d '{ "search_session_id" : [number], "user_session_id" : "useTheSessionIdOfThePreviousCommand", "query" : "deviceVendor = ArcSight", "start_time" : "[yy-mm-ddThh:minutes:seconds.000-end time]", "end_time" : "[yy-mm-ddThh:minutes:seconds.000-end time]}'
```

Uploading CSR using Curl Method

This topic applies to both Software Logger and the Logger Appliance.

This authentication method allows users to generate a CSR (Client Signed Request) using Curl Command. There are 2 ways to upload the certificate using Curl as described below:

To upload the certificate importing a Self Signed Certificate:

1. Log in to Curl to generate the session token

```
curl -k -X GET 'https://<IP or HOST>/core-  
service/rest/LoginService/login?login=<user>&password=<password>' -H 'Accept:  
application/json'
```

2. Sign and load the certificate calling the endpoint

```
curl -k -X POST https://<IP or HOST>/platform-  
service/rest/PlatformService/regenerateSelfSignedCertificate -H 'Content-  
Type: application/xml' -d '<?xml version="1.0" encoding="UTF-8"  
standalone="yes"?> <ns9:regenerateSelfSignedCertificate  
xmlns:ns2="http://ws.v1.service.platform.product.arcsight.com/authenticatio  
nConfigurationService/"  
xmlns:ns3="http://ws.v1.service.platform.product.arcsight.com/entitlementRe  
portService/"  
xmlns:ns4="http://ws.v1.service.platform.product.arcsight.com/managementCli  
Service/"  
xmlns:ns5="http://ws.v1.service.platform.product.arcsight.com/networkConfig  
urationService/"  
xmlns:ns6="http://ws.v1.service.platform.product.arcsight.com/platformConfi  
gurationService/"  
xmlns:ns7="http://ws.v1.service.platform.product.arcsight.com/platformGroup  
Service/"  
xmlns:ns8="http://ws.v1.service.platform.product.arcsight.com/platformLogin  
Service/"  
xmlns:ns9="http://ws.v1.service.platform.product.arcsight.com/platformServi  
ce/"  
xmlns:ns10="http://ws.v1.service.platform.product.arcsight.com/updateServic  
e/"> <ns9:authToken>authTokenString</ns9:authToken>  
<ns9:protocol>HTTPS</ns9:protocol> <ns9:input> <country>US</country>  
<state>California</state> <city>Sunnyvale</city> <organizationName>Micro  
Focus</organizationName> <organizationalUnit>Support  
Team</organizationalUnit> <hostname>example.host.com</hostname>  
<emailAddress>arst-support@microfocus.com</emailAddress>  
<subjectAlternativeName>example.host.com</subjectAlternativeName>
```

```
<keySize>TenTwentyFour</keySize> </ns9:input>  
</ns9:regenerateSelfSignedCertificate>'
```

To upload the certificate generating the CSR

1. Login to Curl to generate the session token

```
curl -k -X GET 'https://<IP or HOST>/core-  
service/rest/LoginService/login?login=<user>&password=<password>' -H 'Accept:  
application/json'
```

2. Generate the CSR

```
curl -k -X POST https://<IP or HOST>/platform-  
service/rest/PlatformService/generateCsr -H 'Content-Type: application/xml'  
-d '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<ns9:generateCsr  
xmlns:ns2="http://ws.v1.service.platform.product.arcsight.com/authenticatio  
nConfigurationService/"  
xmlns:ns3="http://ws.v1.service.platform.product.arcsight.com/entitlementRe  
portService/"  
xmlns:ns4="http://ws.v1.service.platform.product.arcsight.com/managementCli  
Service/"  
xmlns:ns5="http://ws.v1.service.platform.product.arcsight.com/networkConfig  
urationService/"  
xmlns:ns6="http://ws.v1.service.platform.product.arcsight.com/platformConfi  
gurationService/"  
xmlns:ns7="http://ws.v1.service.platform.product.arcsight.com/platformGroup  
Service/"  
xmlns:ns8="http://ws.v1.service.platform.product.arcsight.com/platformLogin  
Service/"  
xmlns:ns9="http://ws.v1.service.platform.product.arcsight.com/platformServi  
ce/"  
xmlns:ns10="http://ws.v1.service.platform.product.arcsight.com/updateServic  
e/"> <ns9:authToken>authTokenString</ns9:authToken>  
<ns9:protocol>HTTPS</ns9:protocol> <ns9:input> <country>US</country>  
<state>California</state> <city>Sunnyvale</city> <organizationName>Micro  
Focus</organizationName> <organizationalUnit>Suppor  
Team</organizationalUnit> <hostname>example.host.com</hostname>  
<emailAddress>arst-support@microfocus.com</emailAddress>  
<subjectAlternativeName>example.host.com</subjectAlternativeName>  
<keySize>TenTwentyFour</keySize> </ns9:input> </ns9:generateCsr>' -o  
server.csr
```

3. Sign the server.csr file

- Generate Root CA by running the following command and enter the information required for the certificate:

```
openssl req -new -x509 -sha256 -newkey rsa:2048 -nodes -keyout ca.key -days 365  
-out ca.crt
```



Note: Common name must be Root_CA.

- Run the following command:

```
openssl x509 -req -in server.csr -days 3650 -sha1 -CAcreateserial -CA  
ca.crt -CAkey ca.key -out server.crt
```

- To verify the generated certificate, run the following three commands:

```
openssl x509 -noout -purpose -in server.crt | grep "SSL server"
```

```
openssl x509 -noout -purpose -in ca.crt | grep "SSL server CA : Yes"
```

```
openssl verify -CAfile ca.crt server.crt
```

4. Upload the file to Logger

- Use base64 command to convert the server.crt to bytes:

```
base64 server.crt
```

- Call the endpoint to upload the certificate. Use the output from the previous step to fill out the base64Output field

```
curl -k -X POST https://<IP or HOST>/platform-  
service/rest/PlatformService/uploadServerCertificate -H 'Content-Type:  
application/xml' -d '<?xml version="1.0" encoding="UTF-8"  
standalone="yes"?><ns9:uploadServerCertificate  
xmlns:ns2="http://ws.v1.service.platform.product.arcsight.com/authentic  
ationConfigurationService/"  
xmlns:ns3="http://ws.v1.service.platform.product.arcsight.com/entitleme  
ntReportService/"  
xmlns:ns4="http://ws.v1.service.platform.product.arcsight.com/managemen  
tCliService/"  
xmlns:ns5="http://ws.v1.service.platform.product.arcsight.com/networkCo  
nfigurationService/"  
xmlns:ns6="http://ws.v1.service.platform.product.arcsight.com/platformC  
onfigurationService/"  
xmlns:ns7="http://ws.v1.service.platform.product.arcsight.com/platformG  
roupService/"  
xmlns:ns8="http://ws.v1.service.platform.product.arcsight.com/platformL  
oginService/"  
xmlns:ns9="http://ws.v1.service.platform.product.arcsight.com/platformS  
ervice/"  
xmlns:ns10="http://ws.v1.service.platform.product.arcsight.com/updateSe  
rvice/"><ns9:authToken>authTokenString</ns9:authToken><ns9:protocol>HTT
```

```
PS</ns9:protocol><ns9:certificate></ns9:certificate>base64Output</ns9:uploadServerCertificate>'
```



Note: Apache server will restart automatically during this process.

5. Confirm upload has been successful



Tip: Verify request commands do not have extra spaces

```
curl -k -X POST https://<IP or HOST>/platform-  
service/rest/PlatformService/getServerCertificateUploadResult -H 'Content-  
Type: application/xml' -d '<?xml version="1.0" encoding="UTF-8"  
standalone="yes"?><ns9:getServerCertificateUploadResult  
xmlns:ns2="http://ws.v1.service.platform.product.arcsight.com/authentication  
nConfigurationService/"  
xmlns:ns3="http://ws.v1.service.platform.product.arcsight.com/entitlementRe  
portService/"  
xmlns:ns4="http://ws.v1.service.platform.product.arcsight.com/managementCli  
Service/"  
xmlns:ns5="http://ws.v1.service.platform.product.arcsight.com/networkConfig  
urationService/"  
xmlns:ns6="http://ws.v1.service.platform.product.arcsight.com/platformConfi  
gurationService/"  
xmlns:ns7="http://ws.v1.service.platform.product.arcsight.com/platformGroup  
Service/"  
xmlns:ns8="http://ws.v1.service.platform.product.arcsight.com/platformLogin  
Service/"  
xmlns:ns9="http://ws.v1.service.platform.product.arcsight.com/platformServi  
ce/"  
xmlns:ns10="http://ws.v1.service.platform.product.arcsight.com/updateServic  
e/"><ns9:authToken>authTokenString</ns9:authToken><ns9:protocol>HTTPS</ns9:  
protocol></ns9:getServerCertificateUploadResult>'
```


Chapter 2: SOAP Login Service

The Login Web service enables you to log in to a Logger and acquire an authentication token that is used for all search and report service calls. Additionally, this service enables you to extend or log out of an existing session, and obtain the version of Web services currently running on your Logger. This section describes following the API calls:

extendSession

```
void extendSession(String authToken)
```

This call extends the session identified by the specified authentication token.

Where `authToken` identifies the Logger session where the query runs.

getVersion

```
String getVersion()
```

This call returns the version of the Web services.

The Web services version is different from the Logger version. For example, for Loggers running 7.3, the Web services version is 1.0.0.0.2.

logout

```
void logout(String authToken)
```

Where `authToken` identifies the Logger session where the query runs.

This call ends a session identified by the authentication token and expires it. The authentication token given here is the one that was established using the `login` call.

login

```
String login(String username, String password, int sessionTimeoutInSeconds)
```

All API calls require you to enter an authentication token that identifies the Logger session where the call will run. This call enables you to log in to a Logger and returns the required authentication token.



Note: Clients must pass the login credentials for the authentication method configured in Logger.

For example, you can set the authentication token in this way:

```
authToken = LoginService.login("username", "password", 600);
```

Where:

- `username` is a user configured on Logger. The user must have the appropriate privileges configured for the actions to be taken using the API calls.
For example, the user must be enabled to **View, run, and schedule reports** for Report folder [Firewall] to run those reports.
- `password` is the password associated with the username.
- `sessionTimeoutInSeconds` is the number of seconds of inactivity after which the login session will end. Regardless of the time you set here, inactive user sessions will time out after the time set in Logger under **System Admin > Authentication Settings > Session Settings > Logout Inactive Session After**. The default session timeout is 15 minutes (900 seconds.)

You can extend an existing session by using the `extendSession` call.

Example:

```
login("username", "password", 300);
```



Caution: After a fresh install or upgrade, manually restart all processes once if encountering login issues.

Chapter 3: SOAP Search Service

This section describes the API calls you can use to perform a search on Logger.

You can run any query that conforms to the required Logger syntax.



Note: The permissions of the SOAP Search Service are those of the user authenticated by the LoginService.login call.

Use the following guidelines when using the Search API:

- The Search API can only return search results that do not contain binary data. If the search results contain binary data, the following exception is generated:

```
"Unexpected EOF; was expecting a close tag for element <ns1:data>"
```

- Searching across peers is not supported. Use the RESTful Search Web service if you need to search across peers.
- If the API receives multiple requests simultaneously, then those searches will take a little longer to decrement.
- Search time field options are not supported in SOAP search service as **logger receipt time** has been set by default. In the RESTful search service, the user has the option to set the parameter "search_time". For more information, see ["search" on page 44](#).

How the Search API Works

The Search API uses an iterator pattern to search and retrieve events. To search for events, you start a search session first using the `startSearch` API call. This call also specifies the query to run. Next, you check if any matches were found using the `hasMoreTuples` API call. If matches are found, you use the `getNextTuples` call to retrieve those events. Once all events have been retrieved or if you have retrieved the events you were searching for, you terminate the search session using the `endSearch` call.

```
String authToken = loginService.login("username", "password", 600);
searchService.startSearch("CEF", System.currentTimeMillis() - 2 * 60 * 60 * 1000,
System.currentTimeMillis(), authToken);

String[] arr = searchService.getHeader(authToken);
for (String str : arr) {
    System.out.println(str);
}

while (searchService.hasMoreTuples(authToken)) {
    Tuple [] tuples = searchService.getNextTuples(10, 600, authToken);
    if (tuples != null) {
        for (Tuple tuple : tuples) {
            System.out.println (tuple.getData());
        }
    }
}

searchService.endSearch(authToken);
loginService.logout(authToken);
```

Returning Specific Fields in Search Results

By default, the Search API returns all fields of matching rows. However, if you need to obtain specific fields and not all, define the fields you need using the `cef` command. Doing so creates the new columns and adds them to the tuple's data array. You can refer to the array, `arr[n]` where `n` is the index location, to obtain specific fields.

The following search query creates two new columns, `name` and `deviceVendor`.

```
ICMP* | cef name, deviceVendor
```

The header format of the search results for this query will be:

```
_rowId _EventTime _raw _PeerName name deviceVendor
```

where:

- `_rowId` is the row identifier
- `_EventTime` is the epoch time in which the event was received by Logger.
- `_raw` contains the raw event data
- `_PeerName` is always local
- `name` and `deviceVendor` are cef-defined fields.



Note: Peer search is not supported for the SOAP search service. To search across peers, use the RESTful search service.

In this case, the first element, `_rowId`, is added to tuple's data array at `arr[0]`. Thus, the new columns, `name` and `deviceVendor`, are added at `arr[4]` and `arr[5]`. You can refer to these array locations to access these fields.

endSearch

```
void endSearch(String authToken)
```

Where `authToken` identifies the Logger session where the query runs.

This call terminates the currently running search session on the Logger identified by the authentication token.

getDataforRowIds

```
String[] getDataforRowIds(String [] rowIds, String authToken)
```

Where:

- `rowIds` is a `String` array of row identifiers.
- `authToken` identifies the Logger session where the query runs.

This call looks up the row IDs passed in as an argument and returns a `String` array of matching raw event data corresponding to the row IDs, in the order they were passed. If a row ID is not found, then the corresponding result contains "null".

You can obtain the Row IDs through search queries. For example, for the search query in ["Returning Specific Fields in Search Results" on page 20](#), the header format of the search results for the query:

```
ICMP* | cef name, deviceVendor
```

was:

```
_rowId _EventTime _raw _PeerName name deviceVendor
```

The `_rowIds` returned by that search query are the ones to use with `getDataforRowIds`, should you need access to the corresponding `_raw` event data.



Note: Some searches, such as `ICMP* | cef name | top 5 name`, do not return the row ID. Instead they return created columns like `name _count`. Results from these searches cannot be passed to this call.

Example:

```
String [] result = searchService.getDataforRowIds(new String[] {"100177-0",  
"invalid"}, authToken);
```

getHeader

```
String getHeader(String authToken)
```

Where `authToken` identifies the Logger session where the query runs.

This call gets the header information that specifies the order in which the fields are returned in the matching events.

getNextTuples

```
Tuple[] getNextTuples(int count, long timeout, String authToken)
```

Where:

- `count` is the number of tuples (rows of matching events or aggregated data) to retrieve in one iteration of this call.
- `timeout` is the time in milliseconds the call waits to receive tuples from Logger. If a tuple is not received within this time, the call terminates.
- `authToken` identifies the Logger session where the query runs.

This call retrieves an array of tuples. Depending on the search query, a tuple might contain rows of matching events or aggregated data. If no data is available at the time the call is made, the return value is `null`.



Note: When using `getNextTuples`, be aware of the following:

- If a search operation is in progress but has not found any matching events yet, the `getNextTuples` might not return any data, even though `hasMoreTuples` call returned a true value.
- The `getHeader` call specifies the order of fields returned in a matching event.

hasMoreTuples

```
boolean hasMoreTuples(String authToken)
```

Where `authToken` identifies the Logger session where the query runs.

This call returns true if the search operation (`startSearch`) is searching for or has found matching events that can be retrieved. Once search finishes on the Logger and no more events remain to be retrieved, this call returns false.

startSearch

```
void startSearch(String queryString, long startTime, long endTime, String authToken)
```

Where:

- `queryString` is any search query that conforms to the syntax Logger expects. For example, Error.
- `startTime` marks the epoch time where the search operation begins scanning.
For example, if you want to specify `startTime` as (`$NOW - 2h`) in Java, enter:
`System.currentTimeMillis() - 2 * 60 * 60 * 1000`.
- `endTime` marks the epoch time where the search operation stops scanning.
For example, if you want to specify `endTime` as (`$Now`) in Java, enter:
`System.currentTimeMillis()`.



Note: If you use `startTime` and `endTime`, make sure the time on the client machine is synchronized with the time on Logger. Otherwise, your search results will contain events that don't match the time range you specified.

- `authToken` identifies the Logger session where the query runs.

This call starts a new search session on Logger identified by the authentication token.

Example: Searching for Events

The following example runs a search on a Logger Appliance L7700 for CEF events received in the last 5 hours, and extracts the name field from the matching events.

The values used in this example:

- Client: Java
- Logger: a Logger appliance with an IP address of 192.0.2.5.
- Search for: CEF events
- Time: last 5 hours
- Filter: extract the name field from the matching events
- Session time out: 600 seconds (10 minutes)

In this example, a login session is established first. If the session does not time out, a search session begins. If matching events are found, they are retrieved, 50 rows at a time, using the `getNextTuples` call. If no rows are returned within 10 minutes of the last retrieval call (`getNextTuples`), the search session terminates. Or, once all rows have been retrieved, the search session ends.



Note: If the following search is run on a software Logger or a Logger Appliance L8000, make sure to add the Logger port number to the `_loggerHost` variable. For example, `192.0.2.5:9000`.

```
package com.coolcustomer.logger.webservices;

import org.apache.axis2.client.Options;
import org.apache.axis2.client.ServiceClient;

import com.arcsight.wsclient.logger.login.adb.LoginServiceStub;
import com.arcsight.wsclient.logger.search.adb.SearchServiceStub;
import com.arcsight.wsclient.logger.search.adb.SearchServiceStub.Tuple;

public class LoggerSearchAPIExample {
    private SearchServiceStub _searchService = null;
    private LoginServiceStub _loginService = null;
    private String _loggerHost = "192.0.2.5";
    private String user = "username";
    private String password = "password";
    private int timeout = 600;

    public String runSearch (String query) {
        init(_loggerHost);

        String authToken = null;
```



```
try {
    String version = _loginService.getVersion();
    System.out.println(version);
    authToken = _loginService.login(user, password, timeout);
    _searchService.startSearch(query,
        System.currentTimeMillis() - (5 * 60 * 60 * 1000),
        System.currentTimeMillis(), authToken);

    // See what the format of the Tuples is
    String [] header = _searchService.getHeader(authToken);
    for (String str : header) {
        System.out.println(str);
    }

    int rowNum = 0;
    while (_searchService.hasMoreTuples(authToken)) {
        Tuple [] tuples =
            _searchService.getNextTuples(50, 600, authToken);
        if (tuples != null) {
            for (Tuple tuple : tuples) {
                String [] arr = tuple.getData();
                System.out.println(" *** Row: " + ++rowNum + " *** ");
                for (int i=0; i<header.length; i++) {
                    System.out.println(arr[i]);
                }
                System.out.println("\n\n");
            }
        }
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    // clean up
    if (authToken != null) {
        try {
            _searchService.endSearch(authToken);
            _loginService.logout(authToken);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
return null;
}

private void init(String loggerHost) {
    XTrustProvider.install();

    try {
        _loginService =
            new LoginServiceStub("https://" + loggerHost +
                "/soap/services/LoginService/LoginService.wsdl");
    }
}
```

```
        _searchService =
            new SearchServiceStub("https://" + loggerHost +
                "/soap/services/SearchService/SearchService.wsdl");

        // read time out from a property file

        // 30 minutes
        long timeOutInMilliseconds = 30 * 60 * 1000;
        Options axisOptions = new Options();
        axisOptions.setTimeoutInMilliseconds(timeOutInMilliseconds);
        ServiceClient serviceClient = _loginService._getServiceClient();
        serviceClient.setOverrideOptions(axisOptions);
        ServiceClient _searchServiceClient =
            _searchService._getServiceClient();
        _searchServiceClient.setOverrideOptions(axisOptions);

    } catch (Exception e) {
        e.printStackTrace();
    }

}

public static void main(String[] args) throws Exception {
    LoggerSearchAPIExample example = new LoggerSearchAPIExample();
    example.runSearch("CEF | cef name");
}
}
```

Chapter 4: SOAP Report Service

This section describes the SOAP Report Web service API calls you can use to run a report on Logger. It covers the following topics:

Some report formats return results in binary format. Therefore, report results are base-64 encoded. You need to decode these results to display them in human-readable form.



Note: The permissions of the SOAP Report Service are those of the user authenticated by the `LoginService.login` call.

getDeviceGroups

```
String[] getDeviceGroups(String authToken)
```

This call returns an array of the names of device groups configured on the Logger that is identified by the specified authentication token.

getDevices

```
String[] getDevices(String authToken)
```

This call returns an array of the names of devices configured on the Logger that is identified by the specified authentication token.

getDevicesInDeviceGroup

```
String[] getDevicesInDeviceGroup(String authToken, String deviceGroupName)
```

This call returns an array of the names of all devices in the specified device group on the Logger that is identified by the specified authentication token.

getReportGroups

```
Group[] getReportGroups(String authToken)
```

This call returns an array of report groups, where each group is associated with a name and a unique report group identifier (`groupID`), on the Logger that is identified by the specified authentication token.

The report groups are the same as report categories in the Logger UI.

getSubGroups

```
Group[] getSubGroups(String groupId, String authToken)
```

This call returns an array of groups within the group whose identifier you specified (`groupId`), on the Logger that is identified by the specified authentication token.


The report groups are the same as report categories in the Logger UI.

getReportsInGroup

```
Report[] getReportsInGroup(String groupId, String authToken)
```

This call returns an array of reports in the specified group (identified by the `groupId`) on the Logger that is identified by the specified authentication token. Each report in the returned array is associated with a report name and a unique report identifier (`reportID`).

The report groups are the same as report categories in the Logger UI.

 **Note:** Use the `getReportGroups` call to obtain `groupId`.

getStorageGroups

```
String[] getStorageGroups(String authToken)
```

This call returns an array of the storage group names configured on the Logger that is identified by the specified authentication token.

runReport

```
String runReport(String reportID, long startTime, long endTime, int scanlimit,  
int resultRowLimit, String devices, String deviceGroups, String storageGroups,  
String reportParameters, String reportformat, String authToken)
```

This call runs the report specified by the `reportID` parameter on the Logger that is identified by the specified authentication token. The report fields are arranged in the CSV format according to the order defined in the report on Logger and are base-64 encoded. You must use a decoder to convert this data into human-readable form. To decode a base-64 encoded report, you need the `ws-commons-util-1.0.1.jar` file.

- `reportID` is a unique identifier for a report. To obtain `reportID`, use `getReportsInGroup` call.
- `startTime` marks the epoch time where the search operation begins scanning.
For example, if you want to specify `startTime` as (`$NOW - 2h`) in Java, enter:
`System.currentTimeMillis() - 2 * 60 * 60 * 1000`.
- `endTime` marks the epoch time where the search operation stops scanning.
For example, if you want to specify `endTime` as (`$Now`) in Java, enter:
`System.currentTimeMillis()`.



Note: If you use `startTime` and `endTime`, make sure the time on the client machine is synchronized with the time on Logger. Otherwise, your search results will contain events that don't match the time range you specified.

- `scanlimit` is the number of events to scan. If you specify 0, all events are scanned.
- `resultRowLimit` is the maximum number of rows of report data to return. If you specify 0, all rows are returned.
- `devices` are the names of devices whose events are scanned for this report. If you do not want to specify device names, enter **null**. In that case, all devices are scanned. To specify multiple devices, enter a comma-separated list that is enclosed in double quotes; for example, “finance-2, internal, dev-server3”. To obtain a list of devices configured on a Logger, use the `getDevices` call.
- `deviceGroups` are the names of device groups whose events are scanned for this report. If you do not want to specify a device group name, enter **null**. In that case, all device groups are scanned. To specify multiple device groups, enter a comma-separated list that is enclosed in double quotes; for example, “finance-servers, sales-servers, dev-servers”.
To obtain a list of device groups configured on a Logger, use the `getDeviceGroups` call.
- `storageGroups` are the names of storage groups whose events are scanned for this report. If you do not want to specify a storage group name, enter **null**. In that case, all storage groups on Logger are scanned. To specify multiple storage groups, enter a comma-separated list that is enclosed in double quotes; for example, “storage-group1, storage-group3”.
To obtain a list of storage groups configured on a Logger, use the `getStorageGroups` call.
- `reportParameters` are the parameters a report requires to run. If a report does not require any parameter, enter **null**. Even if a parameter has default values assigned, those values are not automatically used when a report is run using this API call. You must specify those values in the API call to use them. If a report requires parameters and you do not specify them, the report will not run.

Use double quotes (" ") to separate parameters and single quotes (' ') to separate parameter values.



Note: In Java, double quotes must be escaped by using the backslash (\) character.

- reportFormat is the format in which a report is generated. Only the CSV and PDF formats are supported currently. Enter "CSV" or "csv" for CSV and "PDF" or "pdf" for PDF.
- authToken identifies a session on the Logger on which the report will run. This is a required parameter.

Example: Running a Report

The following example creates a CSV-formatted report that lists the most common events received by a Logger in the last two hours.

The values used in this example:

- Client: Java
- Logger: Logger host logger.companyxyz.com
- Search: Events on Logger host
- Time: Last two hours
- Filter: Display the most common events
- Session time out: 600 seconds (10 minutes)
- Output format: Comma-separated values (CSV) using a base-64 decoder

This example opens a login session first, with a timeout value of 10 minutes.



Note: If the following report is run on a software Logger, make sure to add the Logger port number to the _loggerHost variable. For example, 192.0.2.5:9000.

```
package com.coolcustomer.logger.webservices;

import java.rmi.RemoteException;

import org.apache.axis2.client.Options;
import org.apache.axis2.client.ServiceClient;
import org.apache.ws.commons.util.Base64;

import com.arcsight.wsclient.logger.login.adb.LoginServiceStub;
import com.arcsight.wsclient.logger.report.adb.ArcSightReportServiceException;
import com.arcsight.wsclient.logger.report.adb.ReportServiceStub;

public class LoggerReportAPIExample {
```

```
// LoginService needed to make API calls
private LoginServiceStub _loginService = null;

// ReportService needed to make API calls
private ReportServiceStub _reportService = null;

// IP Address or Hostname (:Port) of the Logger
private String _loggerHost = "192.0.2.5";

private String _login = "username";
private String _password = "password";
private int _timeout = 600;

// Main Method
// A simple test client to run a report by passing in a Name and
// finding the ReportId and running the report
public static void main(String[] args) throws Exception {
    LoggerReportAPIExample example = new LoggerReportAPIExample();
    String result = example.runReport("Most Common Events");
    System.out.println(new String(result));
}

/**
 * This method runs a report, illustrating how to fetch the ID of a
 * report by name
 * @param reportName the name of the report
 * @return result of the report run
 */
public String runReport(String reportName) throws Exception {
    init(_loggerHost);

    // Make a Web service call to login and retrieve an
    //authentication token
    String authToken = _loginService.login(_login, _password,
        _timeout);

    // Fetch the Id for the report from its name, by using a method
    // that recursively loops over all categories and returns the
    // report id
    String id = getReportId(reportName, authToken);

    if (id != null) {
        String result = runReport(id,
            (System.currentTimeMillis() - 2 * 60 * 60 * 1000),
            System.currentTimeMillis(), 0, 100, null, null,
            null, null, "CSV", authToken);
        byte[] reportBytes = Base64.decode(result);
        return new String(reportBytes);
    }

    return null;
}
}
```

```
/**
 * Run a report by passing all the parameters needed by the API
 * @return result of the report run
 * @throws Exception
 */
public String runReport(String reportId, long startTime, long endTime,
    int scanLimit, int resultRowLimit, String devicesCSV,
    String deviceGroupsCSV, String storageGroupsCSV,
    String reportParameters, String reportformat, String
    authToken)
    throws Exception {

    String result = null;

    // Make a Web service call to run the Report
    result = _reportService.runReport(reportId, startTime, endTime,
        scanLimit, resultRowLimit, devicesCSV, deviceGroupsCSV,
        storageGroupsCSV, reportParameters, reportformat,
        authToken);
    return result;
}

/**
 * One way to find a ReportID, is from the Logger Web UI
 * (ReportCategories menu item) Here's a simple programmatic example of
 * how to recurse over the categories to find the report ID
 * @param reportName the name of the report to search for
 * @param authToken the authentication token
 * @return reportID the ID of the report
 * @throws ArcSightReportServiceException
 * @throws RemoteException
 */
private String getReportId(String reportName, String authToken)
    throws ArcSightReportServiceException, RemoteException {

    // get the top level report groups
    ReportServiceStub.Group[] groups = _reportService
        .getReportGroups(authToken);

    for (int i = 0; i < groups.length; i++) {
        ReportServiceStub.Group group = groups[i];

        String groupId = group.getId();
        String groupName = group.getName();

        // Recursively search for the report in all of its subgroups
        String reportId = depthFirstSearchForReport(groupId,
            reportName, authToken);
        if (reportId != null) {
            return reportId;
        }
    }
}
```



```
        return null;
    }

    /**
     * Simple depth first example illustrating the use of the
     * _reportService.getSubGroups method of the API
     * @param groupId the group whose subgroups are needed
     * @param reportName the report that we're looking for recursively
     * @param authToken the authentication token
     * @return reportId, if found
     */
    private String depthFirstSearchForReport(String groupId, String
        reportName,
        String authToken) throws ArcSightReportServiceException,
        RemoteException {

        // get the reports
        ReportServiceStub.Report[] reports = _reportService.getReportsInGroup(
            groupId, authToken);
        if (reports != null) {
            for (int j = 0; j < reports.length; j++) {
                ReportServiceStub.Report report = reports[j];
                String reportID = report.getId();
                if (reportName.equals(report.getName())) {
                    return reportID;
                }
            }
        }

        // if not found here, start recursing over the subgroups
        ReportServiceStub.Group[] subgroups = _reportService.getSubGroups(
            groupId, authToken);
        if (subgroups != null && subgroups.length > 0) {
            for (int i = 0; i < subgroups.length; i++) {
                String subGroupID = subgroups[i].getId();
                String reportId = depthFirstSearchForReport(subGroupID,
                    reportName, authToken);
                if (reportId != null) {
                    return reportId;
                }
            }
        }

        return null;
    }

    private void init(String loggerHost) {
        // Use this class, to make the JRE trust the certificates
        XTrustProvider.install();
        if (_reportService != null && _loginService != null) {
            return;
        }
    }
}
```

```
// Setup the LoginService & ReportService stubs to make API
// calls to your Logger
try {
    _reportService = new ReportServiceStub("https://" + loggerHost
        + "/soap/services/ReportService/ReportService.wsdl");
    _loginService = new LoginServiceStub("https://" + loggerHost
        + "/soap/services/LoginService/LoginService.wsdl");

    // 30 minutes
    long timeOutInMilliseconds = 30 * 60 * 1000;

    // Axis related settings
    Options axisOptions = new Options();
    axisOptions.setTimeoutInMilliseconds(timeOutInMilliseconds);
    ServiceClient serviceClient = _reportService._getServiceClient();
    serviceClient.setOverrideOptions(axisOptions);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Example: Passing Parameters when Running a Report

Before you can run a report through the API, the report must be set up. For this example, we will set up a report that allows users to select from a list of products and a list of vendors.

To create the example report:

1. In the in the **Parameter Object Editor**, create two multi-select lists with predefined values.
 - Multi-Select List 1: commonProducts, with the values **Logger** and **ESM**
 - Multi-Select List 2: commonVendors, with the values **Arcsight**, **Cisco**, and **Juniper**
2. Create the following query in the **Query Object Editor**:

```
SELECT arc_name, arc_deviceProduct, arc_deviceVendor
FROM events
Where lower(arc_deviceProduct) IN (<%commonProducts%>)
OR lower(arc_deviceVendor) IN (<%commomVendors%>)
GROUP BY arc_name, arc_deviceProduct, arc_deviceVendor
LIMIT 5
```

3. In the **Adhoc Report Designer**, create a report called Product_Vendor_Option_Report.
4. Add the query that you just created to the report.

5. Add the Common Products and Common Vendors multi-select lists you just created to this report.

When running the report, users are prompted to enter the parameters based on the query. Logger builds the query based on the user's specification.

For example, if the user selects **Logger** for commonProducts and **Arcsight** and **Cisco** for commonVendors, Logger will fill in the fields like this when running the query:

```
SELECT arc_name, arc_deviceProduct, arc_deviceVendor
FROM events
Where lower(arc_deviceProduct) IN ("logger")
OR lower(arc_deviceVendor) IN ("arcsight", "cisco")
GROUP BY arc_name, arc_deviceProduct, arc_deviceVendor
LIMIT 5
```

In order to run the report through the API, you must pass the parameters that a user would have selected.

To run the example report from the API:

1. Find the Report ID, either using the API or from the UI.

Open **Reports > Report Categories > Deploy Reports and Categories** and note the report ID of the report you want to use.

For example, suppose the Product_Vendor_Option_Report that we created has the Report ID 1C568A25-8458-50E1-2C7E-7605291C5EB4.

2. Run the report from the API as follows:

```
String result = reportService.runReport(
    "1C568A25-8458-50E1-2C7E-7605291C5EB4", // Report ID
    System.currentTimeMillis() - 60 * 60 * 1000, // Start Time
    System.currentTimeMillis(), // End Time
    10000, 100, null, // Scan Limit, Row Limit, Devices
    null, null, // Device Groups, Storage Groups
    "\"commonVendors='arcsight', 'cisco'\", \"commonProducts='logger'\"",
    // Comma Separated parameters
    "csv", // Output Format
    authToken); // Authentication token
byte[] data = Base64.decode(result);
String reportResult = new String(data);
```

Be sure to use quotes to identify comma-separated parameter strings. In this example, the string that needs to be sent is:

```
"commonVendors='arcsight', 'cisco'", "commonProducts='logger'"
```

In Java, the quotes must be escaped by using the backslash (\) character, like this:

```
String str = "\"commonVendors='arcsight', 'cisco'\",  
\"commonProducts='logger'\"";
```

Chapter 5: Logger's RESTful Web Services

This section provides information that applies to all of Logger's RESTful Web services. It covers the following topics:

About Logger's RESTful Web Services

The following RESTful Web service APIs are included.

- **Login Service:** For more information, see ["RESTful Login Service" on page 40](#).
- **Search Service:** For more information, see ["RESTful Search Service" on page 43](#).



Tip: RESTful services for reports are not currently supported. For reporting, use the SOAP Web service, described in ["SOAP Report Service" on page 27](#).

Error Messages

If a RESTful Search API returns an error, the error message will be in the following format:

```
{"errors": [{"message": "<Information about error>", "code": <error code>}]}
```

The range of error codes for the restful Search APIs is 1000-1999.

Example Search API Error Messages

```
{"errors": [{"message": "No search session id is provided", "code": 1003}]}  
Response status code: 400
```

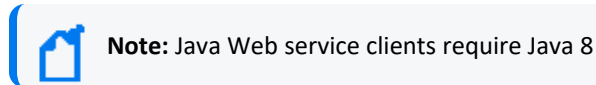
```
{"errors": [{"message": "User session OmyExT3o0Ca8zINR4X_3VW3YiXmh_jTpXSI8H7XIg4. is not  
valid", "code": 1002}]}  
Response status code: 401
```

```
{"errors": [{"message": "License status is updated", "code": 1004}]}  
Response status code: 401
```

```
{"errors": [{"message": "Invalid license: The signature in the license file is invalid.  
Your license file is corrupt. Please contact ArcSight Customer Support.",  
"code": 1005}]}  
Response status code: 401
```

Supported REST Web Service Clients

While most of the REST examples in this document use curl, you could use a Web service client you wrote, or a standard REST client instead, as illustrated in the following examples. Both examples below use the login request to authenticate a user and, if successful, return a session ID. For more information on login requests, see ["login" on page 40](#).



Example: Making a Login Request Using Perl Script

```
#!/usr/bin/perl
use strict;
use warnings;

my $host = $ARGV[0] || die "usage: $0 hostname\n";
my $login="username";
my $password="password";

#
# get the authToken
#
my $cmd = "curl -H 'Accept: application/json' -X POST -d
'login=$login&password=$password' --insecure 'https://$host/core-
service/rest/LoginService/login' 2>&1";

my $s = `$cmd`;
my ($authToken) = $s =~ /log.return":"([\_-\w\d\.]+)"/xms;
die "failed to acquire authToken\n" unless $authToken;

print $authToken, "\n";
```

- If the invocation is successful:

```
# ./login.pl <hostname>
UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.
```

- If the invocation is unsuccessful:

```
# ./login.pl <hostname>
failed to acquire authToken
```

Example: Making a Login request using Curl

```
curl -X POST -d "login=username&password=password" -k "  
https://15.214.132.82:9000/core-service/rest/LoginService/login"  
  
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
  
<ns3:loginResponse  
xmlns:ns2="http://ws.v1.service.core.product.arcsight.com/groupService/"  
xmlns:ns3="http://ws.v1.service.core.product.arcsight.com/loginService/"  
xmlns:ns4="http://ws.v1.service.core.product.arcsight.com/userService/">  
  
<ns3:return>UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.</ns3:return>  
  
</ns3:loginResponse>
```

Chapter 6: RESTful Login Service

This section describes the RESTful Login Web service that you can use to authenticate a user, log into Logger, and establish a session ID to use When making API calls.

The RESTful Login Service supports HTTP GET and POST requests described below.

Micro Focus recommends that you use POST instead of GET when making a call to the REST login() API, so that the username and password are not written to the Apache logs.

You should use GET for testing purposes only. When using GET, be aware that:

- The URL parameters (such as username and password) will be written to the Apache logs.
- The URL parameters must be url-encoded.

login

All REST calls require a User Session ID. This call provides user authentication and opens the session. It returns a User Session ID (in simple XML format) for you to use when making REST calls to Logger during this session.



Note: Use the login credentials for the authentication method configured in Logger.

Resource URL

Use the following URL when making Login requests.

```
https://<hostname>:<port>/core-service/rest/LoginService/login
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
login	String	Yes	The Logger user's login ID This user must already exist on Logger.
password	String	Yes	The Logger user's password

Response

This request returns one of the following status codes.

Status Code	Description
200	Request completed successfully.
500	Request error or authentication failure.

This request returns the following values.

Attribute	Description
(XML)	The User Session ID to use in request operations.

Example Login Request

This curl example authenticates a user and returns a User Session ID.

```
curl -X POST -d "login=username&password=password" -k "
https://15.214.132.82:9000/core-service/rest/LoginService/login"

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:loginResponse
  xmlns:ns2="http://ws.v1.service.core.product.arcsight.com/groupService/"
  xmlns:ns3="http://ws.v1.service.core.product.arcsight.com/loginService/"
  xmlns:ns4="http://ws.v1.service.core.product.arcsight.com/userService/">
<ns3:return>UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.</ns3:return>
</ns3:loginResponse>
```

logout

This call logs out the user and closes the user session. The User Session ID generated previously will no longer be available.

Resource URL

Use the following URL when making Logout requests.

```
https://<hostname>:<port>/core-service/rest/LoginService/logout
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
authToken	String	Yes	The User Session ID

Response

This request returns one of the following status codes.

Status Code	Description
200	Request completed successfully.
500	Request error or the User Session ID provided was invalid/expired.

This request returns the following values.

Attribute	Description
(XML)	No values in the returned XML.

Example Logout Request

This example logs out of and closes the User Session ID UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.

```
https://<hostname>:<port>/core-service/rest/LoginService/logout?authToken=UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns3:logoutResponse
  xmlns:ns2="http://ws.v1.service.core.product.arcsight.com/groupService/"
  xmlns:ns3="http://ws.v1.service.core.product.arcsight.com/loginService/"
  xmlns:ns4="http://ws.v1.service.core.product.arcsight.com/userService/">
```

Chapter 7: RESTful Search Service

This section describes the RESTful Search Web service that you can use to search events stored in Logger.

The RESTful Search Web service supports the HTTP POST requests described below.

Keep the following in mind when making search requests.

- Unlike the SOAP Search Web service, you can use the RESTful Search Web service to run distributed searches.
- The request body and responses are in JSON (JavaScript Object Notation) format.



Note: The permissions of the RESTful Search Service are those of the user authenticated by the LoginService/login call.

http status codes

Any search request could return one of the following status codes. Additional codes that may be returned are listed under each operation, where applicable.

Status Code	Description
2XX	Request completed successfully.
400	Request error. Invalid JSON request or parameter. See body of the response for details.
401	Invalid User Session ID or license, user has no search permission. See body of the response for details.
404	Requested search session is not found or the requested REST API does not exist.
500	Unspecified internal server error. See body of the response for details.

date/time format

Use the following ISO-8601 compliant date/time format in request parameters.

yyyy-MM-dd'T'HH:mm:ss.SSSXXX

For example, May 26 2014 at 21:49:46 PM could have a format like one of the following:

- In PDT: 2014-05-26T21:49:46.000-07:00
- In UTC: 2014-05-26T21:49:46.000Z

Code	Description
yyyy	Four digit year
MM	Two-digit month (01=January, etc.)
dd	Two-digit day of month (01 through 31)
T	Separator for date/time
HH	Two digits of hour (00 through 23) (am/pm NOT allowed)
mm	Two digits of minute (00 through 59)
ss	Two digits of second (00 through 59)
SSS	Three digit milliseconds of the second
XXX	ISO 8601 time zone (Z or +hh:mm or -hh:mm)

search

Starts a new search.

Resource URL

Use the following URL when making search requests.

```
https://<hostname>:<port>/server/search
```



Note: If your query string includes special characters, use standard URI encoding. In that case, add the parameter "uri_encoded": true.

Parameters

This request accepts the following parameters.

Name	Type	Required	Default	Description
search_session_id	Number	Yes		The Search Session ID to be used in future search related request operations. This must be an increasing positive integer. (For example, you could use the server time in milliseconds.)
user_session_id	String	Yes		The User Session ID generated by the login API.
discover_fields	Boolean		false	It indicates that the search should try to discover fields in the events found. Will be considered when field_summary=true. Otherwise, ignored.
end_time	String			A string defining the end date and time of the search. See "date/time format" on page 43 for the format. Please note that dynamic dates can be used. If end_time is provided, start_time needs to be present as well.
summary_fields	Array of String		["Event Time", "Device", "Logger", "Raw Message", "deviceVendor", "deviceProduct", "deviceVersion", "deviceEventClassId", "name"]	The list of fields (display name, not CEF) in a array to be used to calculate summary when field_summary is true.
field_summary	Boolean		false	Indicates to use the field summary.
local_search	Boolean		true	It indicates the search is local only, and does not include peers. Set to false if you want to include peers in the search.

Name	Type	Required	Default	Description
query	String		"" (null string)	<p>The search query string to filter/process the events.</p> <p>No control characters are allowed in the query parameter.</p> <p>The escape character for double quotes (") and backslashes (\) in the query is the backslash.</p> <ul style="list-style-type: none"> To include a double quote in your query, use \". To include a backslash in your query, use \. <p>If you query include aggregate operators such as sort, tail or head, refer to "chart_data" on page 55.</p>
search_type	String		interactive	<p>The search type. Only the default value, interactive, is supported. Interactive searches send a query to the server and return the query output.</p>
start_time	String		2 hours	<p>A string defining the beginning date and time of the search. See "date/time format" on page 43 for the format. Please note that dynamic dates can be used.</p> <p>If start_time is provided, end_time needs to be present as well.</p>
timeout	Number		120000	<p>The number of milliseconds to keep the search after processing has stopped.</p> <p>Note: This timeout is only two minutes. If you need to keep the search longer, increase this number.</p>
search_time	string		"received_time"	<p>There are 2 options: "event_time" and "received_time"</p> <p>It indicates the field date used for searching events.</p>

Response

This request returns the following status code or one of the status codes listed in "[http status codes](#)" on page 43.

Status Code	Description
409	Failed to create a new search.

This request returns the following values.

Attribute	Description
sessionId	Server session ID. In Logger, you can use this session ID to identify and stop the search on Running Tasks page.

For information about returned error messages, see "[Error Messages](#)" on page 37.

Status

Returns the latest status of the specified search.

Resource URL

Use the following URL when making status requests.

```
https://<hostname>:<port>/server/search/status
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.

Response

This request returns one of the status codes listed in "[http status codes](#)" on page 43.

This request returns the following values.

Attribute	Description
elapsed	The elapsed time of this search in format HH:mm:ss.SSS. If the specified search is not started yet, 0 will be returned.
hit	The number of events found. If the specified search is not started yet, 0 will be returned.
message	The message for the search. This will be used when there is an error in the local/peer search.
result_type	Indicates the type of the search results. When the search results have a chart, it returns "chart"; when search has "sort, tail or head" operators, it returns "aggregate"; otherwise, it returns "histogram". If the result type is aggregate, you need to use the chart_data operator to return the results.
scanned	The number of events scanned. If the specified search is not started yet, 0 will be returned.
status	The search status. Can be any of starting, running, complete, or error. If the specified search has not started yet, the status will be "starting".
startTimeSearch	The time when the search started.
endTimeSearch	The time when the search ended.
elapsedMillis	The elapsed time of the search process in milliseconds.

For information about returned error messages, see "[Error Messages](#)" on page 37.

Example

This example returns that the status of the search session 1399546550086 performed by the user session UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw is complete.

```
curl -k https://<hostname>:<port>/server/search/status -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."
}'
{
  "status": "running",
  "result_type": "histogram",
  "hit": 1791,
  "scanned": 1791,
  "startTimeSearch": "2020/06/09 07:38:31 PDT",
  "endTimeSearch": "2020/06/11 07:38:31 PDT",
```



```
}      "elapsedMillis": 13285,  
      "elapsed": "00:00:13.285",  
      "message": []  
}
```

histogram

Returns data you can use to display a histogram (a column chart with no gap between columns) of the event distribution over time of an already searched time range.

Resource URL

Use the following URL when making histogram requests.

```
https://<hostname>:<port>/server/search/histogram
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.

Response

This request returns one of the status codes listed in "[http status codes](#)" on page 43.

This request returns one of the following values.

Attribute	Description
bucket_count	The number of buckets in the histogram.
bucket_width	The bucket width or unit in millisecond.
hits	The list of hit event count for each bucket.
start_bucket_time	The start bucket time or left most time of the buckets in epoch time. When bucket_count is 0, this value is 0.

For information about returned error messages, see "[Error Messages](#)" on page 37.

Example

This example returns data you can use to display a histogram.

```
curl -k https://<hostname>:<port>/server/search/histogram -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."
}'

{
  "bucket_count" : 25,
  "bucket_width" : 60000,
  "hits" : [173, 190, 190, 0, 0, 0, 0, 42, 190, 173, 133, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 52, 10],
  "start_bucket_time" : 1399243148000,
}
```

drilldown

Narrows the search results to the specified time range. For example, you can use it to narrow down the search results to be shown in the grid when a bar of the histogram is clicked.

Resource URL

Use the following URL when making drilldown requests.

```
https://<hostname>:<port>/server/search/drilldown
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.

Name	Type	Required	Description
end_time	String	Yes	A string defining the end date and time of the search. See "date/time format" on page 43 for the format. If end_time is provided, start_time needs to be present as well.
start_time	String	Yes	A string defining the beginning date and time of the search. See "date/time format" on page 43 for the format. If start_time is provided, end_time needs to be present as well.

Response

This request returns no values, only one of the status codes listed in [" http status codes" on page 43](#).

For information about returned error messages, see ["Error Messages" on page 37](#).

Example

This example narrows down the search results to the time range from 04/10/2014 10 am to 12 pm in PDT.

```
curl -k https://<hostname>:<port>/server/search/drilldown -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.",
  "start_time" : "2014-04-10T10:00:00.000-07:00",
  "end_time" : "2014-04-10T12:00:00.000-07:00"
}'
```

events

Returns the list of events found in the specified search.

Resource URL

Use the following URL when making events requests.

```
https://<hostname>:<port>/server/search/events
```

Parameters

This request accepts the following parameters.

Name	Type	Required	Default	Description
search_session_id	Number	Yes		The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes		The User Session ID generated by the login API.
dir	String		forward	The sort direction based on event time. forward/backward
fields	String		"" (null string)	The list of fields in the order to show. If not specified, all fields will be used
length	Number		1000	The length or number of events to retrieve. Maximum number is 10000.
offset	Number		0	The offset from the first event.
userDefined	Boolean		False	The userDefined establish if the field sets are specified.

Response

This request returns one of the status codes listed in "[http status codes](#)" on page 43.

This request returns the following values.

Attribute	Description
fields	<p>The list of field objects for the results. If the fields are specified in the request or when starting this search, the field names and the order will be same as specified.</p> <p>The field object will have:</p> <ul style="list-style-type: none">• name: Field name• type: Field type (string/number/date)• alias: Original name if the field name is renamed. <p>If name starts with an underscore "_", then it is an internal field and not for displaying.</p>
results	<p>The list of results or values from the events found. This will be list of arrays, where each array has values for each field specified in the fields attribute.</p>

For information about returned error messages, see "[Error Messages](#)" on page 37.

Example

This example returns a list of events including the specified fields only.

```
curl -k https://<hostname>:<port>/server/search/events -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.",
  "fields" : ["deviceEventClassId", "destinationAddress", "deviceVendor",
"deviceReceiptTime", "endTime", "baseEventCount", "deviceAddress"]
}'

{
  "fields" : [
    {"name": "_rowId", "type": "string", "alias": "_rowId"},
    {"name": "deviceEventClassId", "type": "string", "alias":
"deviceEventClassId"},
    {"name": "destinationAddress", "type": "string", "alias":
"destinationAddress"},
    {"name": "deviceVendor", "type": "string", "alias": "deviceVendor"},
    {"name": "deviceReceiptTime", "type": "date", "alias":
"deviceReceiptTime"},
    {"name": "endTime", "type": "date", "alias": "endTime"},
    {"name": "baseEventCount", "type": "number", "alias":
"baseEventCount"},
    {"name": "deviceAddress", "type": "string", "alias": "deviceAddress"}
  ],
  "results": [
    ["3E8-0@Local", "TCP_NC_MISS", "192.0.2.1", " Blue Coat",
1277888507046, 1277888507046, 1, "192.0.2.9"],
    ["3E8-1@Local", "TCP_NC_MISS", "192.0.2.1", " Blue Coat",
1277888507046, 1277888507046, 1, "192.0.2.9"],
    ...
  ]
}
```

raw_events

Returns the raw events for the specified row IDs.

Resource URL

Use the following URL when making raw events requests.

```
https://<hostname>:<port>/server/search/raw_events
```

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.
row_ids	Array of String	Yes	The list of row IDs to retrieve the raw events from the search results.

Response

This request returns one of the status codes listed in "[http status codes](#)" on page 43.

This request returns the following values.

Attribute	Description
(JSON Array)	The array of raw events specified by the row IDs.

For information about returned error messages, see "[Error Messages](#)" on page 37.

Example

This example returns the raw data in the specified rows.

```
curl -k https://<hostname>:<port>/server/search/raw_events -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.",
  "row_ids" : ["3FA84-0"]
}'

{
  ["CEF:0|ArcSight|Logger|6.0.0.0.0|storagegroup:100|Storage Group Space Used|1| cat=/Monitor/StorageGroup/Space/Used cn1=1 cn1Label=Percent Used cn2=180 cn2Label=retention period (days) cn3=1024 cn3Label=used (MB) cs2=CurrentValue cs2Label=timeframe dst=192.0.2.18 dvc=192.0.2.18 end=1399546170515 fileType=storageGroup fname=Default Storage Group fsize=71
```

```
rt=1399546170515"]  
}
```

chart_data

Returns the data you can use to display a chart and the table under the chart. The `chart_data` request also returns the results of aggregate operators like `sort`, `tail`, and `head`. For an example of returning the results of aggregate operators, see ["Example: Returning Aggregate Search Data" on page 61](#).

Resource URL

Use the following URL when making chart data requests.

```
https://<hostname>:<port>/server/search/chart_data
```



Note: In order to get valid `chart_data` results, the search query you made earlier must include the pipeline chart operator. For example:

```
... |chart sum(deviceCustomNumber1) by deviceEventClassId
```

Parameters

This request accepts the following parameters.

Name	Type	Required	Default	Description
search_session_id	Number	Yes		The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes		The User Session ID generated by the login API.
length	Number		25	The length or number of results to retrieve. Maximum number is 100.
offset	Number		0	The offset from the first result.

Response

This request returns one of the status codes listed in "[http status codes](#)" on page 43.

This request returns the following values.

Attribute	Description
fields	The list of field objects for the results. The field object will have: "name": field name, "type": field type (string/number/date), "alias": original name if the field name is renamed.
results	The list of results for each field. This will be list of arrays, where each array has values for each field specified in the fields attribute.

For information about returned error messages, see "[Error Messages](#)" on page 37.

Example 1

This example returns the data necessary to display a chart.

```
curl -k https://<hostname>:<port>/server/search/chart_data -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."
}'

{
  "fields" : [
    {"name": "deviceEventClassId", "type": "string", "alias": "deviceEventClassId"},
    {"name": "sum_deviceCustomNumber1", "type": "number", "alias": "sum_deviceCustomNumber1"}
  ],
  "results": [
    ["TCP_NC_MISS", 450]
    ...
  ]
}
```

stop

Stops the search operation but keeps the search session so that the search results can be narrowed down later.

Resource URL

Use the following URL when making stop requests.

```
https://<hostname>:<port>/server/search/stop
```



Note: The data already returned is stored on the server until the timeout specified in the search operation has been reached.

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.

Response

This request returns no values, only one of the status codes listed in "[http status codes](#)" on [page 43](#).

For information about returned error messages, see "[Error Messages](#)" on [page 37](#).

Example

This example stops the search session 1399546550086 performed by user session UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.

```
curl -k https://<hostname>:<port>/server/search/stop -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."
}'
```

close

Stops the execution of the search and clears the search session data from the server.

Resource URL

Use the following URL when making close requests.

`https://<hostname>:<port>/server/search/close`

Parameters

This request accepts the following parameters, with no defaults.

Name	Type	Required	Description
search_session_id	Number	Yes	The Search Session ID you specified when first starting the search session.
user_session_id	String	Yes	The User Session ID generated by the login API.

Response

This request returns no values, only one of the status codes listed in "[http status codes](#)" on [page 43](#).

For information about returned error messages, see "[Error Messages](#)" on [page 37](#).

Example

This example stops the search session 1399546550086 performed by user session UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.

```
curl -k https://<hostname>:<port>/server/search/close -H "Content-Type: application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."
}'
```

Example: Running a RESTful Search

This example demonstrates the steps you need to follow to run a RESTful API search, from start to finish. It includes logging in, opening a search session, getting a list of events, closing the search session, and logging out.

Step 1: Log In and Get a User Session ID

Use the returned User Session ID in all requests in the rest of the user session.

```
https://<hostname>:<port>/core-service/rest/LoginService/login?login=username&
password=password

<ns3:loginResponse
xmlns:ns2="http://ws.v1.service.core.product.arcsight.com/groupService/"
xmlns:ns3="http://ws.v1.service.core.product.arcsight.com/loginService/"
xmlns:ns4="http://ws.v1.service.core.product.arcsight.com/userService/">
  <ns3:return>UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.</ns3:return>
</ns3:loginResponse>
```

Step 2: Open a Search Session

Use the Search Session ID you specify here in all request in the rest of the session. You can have more than one Search Session per User session.

```
curl -k https://<hostname>:<port>/server/search -H "Content-Type: application/json ;
charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.",
  "query" : "_deviceGroup IN [\"Logger Internal Event Device [cef_events]\"]",
  "start_time" : "2014-04-02T22:08:44.000-07:00",
  "end_time" : "2014-05-02T22:08:44.000-07:00",
  "field_summary":true
}'

{
  "sessionId" : "104857600"
}
```

Step 3: Request the Desired Data

This example returns a list of events. You could make other calls such as status or histogram instead or as well.

```
curl -k https://<hostname>:<port>/server/search/events -H "Content-Type:
application/json ; charset=UTF-8" -d '{
  "search_session_id" : 1399546550086,
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.",
  "fields" : ["deviceEventClassId", "destinationAddress", "deviceVendor",
"deviceReceiptTime", "endTime", "baseEventCount", "deviceAddress"]
}'

{
  "fields" : [
    { "name": "_rowId", "type": "string", "alias": "_rowId"},
    { "name": "deviceEventClassId", "type": "string", "alias":
"deviceEventClassId"},
```

```
    {"name": "destinationAddress", "type": "string", "alias":  
"destinationAddress"},  
    {"name": "deviceVendor", "type": "string", "alias": "deviceVendor"},  
    {"name": "deviceReceiptTime", "type": "date", "alias":  
"deviceReceiptTime"},  
    {"name": "endTime", "type": "date", "alias": "endTime"},  
    {"name": "baseEventCount", "type": "number", "alias": "baseEventCount"},  
    {"name": "deviceAddress", "type": "string", "alias": "deviceAddress" }  
  ],  
  "results": [  
    ["3E8-0@Local", "TCP_NC_MISS", "192.0.2.1", " Blue Coat", 1277888507046,  
1277888507046, 1, "192.0.2.9"],  
    ["3E8-1@Local", "TCP_NC_MISS", "192.0.2.1", " Blue Coat", 1277888507046,  
1277888507046, 1, "192.0.2.9"],  
    ...  
  ]  
}
```

Step 4: Close the Search Session

To identify the search session to close, use the Search Session ID and the User Session ID. After this, you can log out of the user session or open another one.

```
curl -k https://<hostname>:<port>/server/search/close -H "Content-Type:  
application/json ; charset=UTF-8" -d '{  
  "search_session_id" : 1399546550086,  
  "user_session_id" : "UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw."  
}'
```

Step 5: Log Out of the User Session

To end the user session, use the User Session ID as the auth token.

```
https://<hostname>:<port>/core-service/rest/LoginService/logout?authToken=  
UDIWj3m-iGksVI_zSMViSdqF48r6DXpbTQpRQQiEbgw.
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<ns3:logoutResponse  
xmlns:ns2="http://ws.v1.service.core.product.arcsight.com/groupService/"  
xmlns:ns3="http://ws.v1.service.core.product.arcsight.com/loginService/"  
xmlns:ns4="http://ws.v1.service.core.product.arcsight.com/userService/">
```

Example: Returning Aggregate Search Data

This example demonstrates how to return aggregate search data from the sort, tail, and head operators.

Query:

```
curl -k https://15.214.134.204:9000/server/search -H "Content-Type:application/json ; charset=UTF-8" -d '{
  "search_session_id":1399546550086,
  "user_session_id":"C9ULaqqTtpk9-oGcrr6NoQ8iHbrbuGNlWtP9S7JkpxE.",
  "query":"_deviceGroup in [\"Logger Internal Event Device\"] | sort Time",
  "start_time":"2016-06-29T22:08:44.000-07:00",
  "end_time":"2016-06-29T22:09:00.050-07:00",
  "field_summary":true
}'
```

Response:

```
{
  "sessionId":"15"
}
```

Check Status of Query:

```
curl -k httpcurl -k https://15.214.134.204:9000/server/search/status -H "Content-Type:application/json; charset=UTF-8" -d '{
  "search_session_id":1399546550086,
  "user_session_id":"C9ULaqqTtpk9-oGcrr6NoQ8iHbrbuGNlWtP9S7JkpxE."
}'
```

Response:

```
{
  "status":"complete",
  "result_type":"aggregate",
  "hit":4,
  "scanned":200,
  "elapsed":"00:00:00.286",
  "message":[
  ]
}
```

Get Events:

```
curl -k https://15.214.134.204:9000/server/search/chart_data -H "Content-Type:application/json;charset=UTF-8" -d '{
  "search_session_id":1399546550086,
  "user_session_id":"C9ULaqqTtpk9-oGcrr6NoQ8iHbrbuGNlWtP9S7JkpxE."
}'
```

Response:

```
{
  "fields":[
    {
      "name": "_rowId",
      "type": "string",
      "alias": "_rowId"
    },
    {
      "name": "Event Time",
      "type": "date",
      "alias": "Event Time"
    },
    {
      "name": "Logger",
      "type": "string",
      "alias": "Logger"
    },
    {
      "name": "Device",
      "type": "string",
      "alias": "Device"
    },
    {
      "name": "Receipt Time",
      "type": "date",
      "alias": "Receipt Time"
    },
    {
      "name": "Time",
      "type": "string",
      "alias": "Time"
    },
    {
      "name": "deviceReceiptTime",
      "type": "date",
      "alias": "deviceReceiptTime"
    }
  ]
}
```

```
    "name": "deviceCustomString2",
    "type": "string",
    "alias": "deviceCustomString2"
  },
  {
    "name": "destinationAddress",
    "type": "string",
    "alias": "destinationAddress"
  },
  {
    "name": "deviceCustomNumber1",
    "type": "number",
    "alias": "deviceCustomNumber1"
  },
  {
    "name": "baseEventCount",
    "type": "number",
    "alias": "baseEventCount"
  },
  {
    "name": "startTime",
    "type": "date",
    "alias": "startTime"
  },
  {
    "name": "deviceVersion",
    "type": "string",
    "alias": "deviceVersion"
  },
  {
    "name": "agentSeverity",
    "type": "string",
    "alias": "agentSeverity"
  },
  {
    "name": "name",
    "type": "string",
    "alias": "name"
  },
  {
    "name": "deviceAddress",
    "type": "string",
    "alias": "deviceAddress"
  },
  {
    "name": "deviceVendor",
    "type": "string",
    "alias": "deviceVendor"
  },
  {
    {
```

```
    "name": "Version",
    "type": "string",
    "alias": "Version"
  },
  {
    "name": "deviceCustomNumber1Label",
    "type": "string",
    "alias": "deviceCustomNumber1Label"
  },
  {
    "name": "deviceEventCategory",
    "type": "string",
    "alias": "deviceEventCategory"
  },
  {
    "name": "deviceProduct",
    "type": "string",
    "alias": "deviceProduct"
  },
  {
    "name": "deviceEventClassId",
    "type": "string",
    "alias": "deviceEventClassId"
  },
  {
    "name": "endTime",
    "type": "date",
    "alias": "endTime"
  },
  {
    "name": "deviceCustomString2Label",
    "type": "string",
    "alias": "deviceCustomString2Label"
  },
  {
    "name": "deviceCustomString1",
    "type": "string",
    "alias": "deviceCustomString1"
  },
  {
    "name": "deviceCustomString3",
    "type": "string",
    "alias": "deviceCustomString3"
  },
  {
    "name": "deviceCustomString6",
    "type": "string",
    "alias": "deviceCustomString6"
  },
  {
```



```
    "name": "deviceCustomString1Label",
    "type": "string",
    "alias": "deviceCustomString1Label"
  },
  {
    "name": "deviceCustomString6Label",
    "type": "string",
    "alias": "deviceCustomString6Label"
  },
  {
    "name": "deviceCustomString3Label",
    "type": "string",
    "alias": "deviceCustomString3Label"
  },
  {
    "name": "deviceCustomNumber2",
    "type": "number",
    "alias": "deviceCustomNumber2"
  },
  {
    "name": "deviceCustomNumber2Label",
    "type": "string",
    "alias": "deviceCustomNumber2Label"
  }
],
"results": [
  [
    "2E047-28@Local",
    1467263340015,
    "Local",
    "Logger",
    1467263789004,
    "",
    1467263340015,
    "CurrentValue",
    "127.0.0.1",
    1,
    1,
    1467263340015,
    "6.3.0.30273.0",
    "1",
    "CPU Usage",
    "127.0.0.1",
    "ArcSight",
    "0",
    "Percent Usage",
    "/Monitor/CPU/Usage",
    "Logger",
    "cpu:100",
    1467263340015,
```

```
    "timeframe",
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    null,
    ""
  ],
  [
    "2E047-29@Local",
    1467263340016,
    "Local",
    "Logger",
    1467263789004,
    "",
    1467263340016,
    "SinceLastMonitorEvent",
    "127.0.0.1",
    5,
    1,
    1467263340016,
    "6.3.0.30273.0",
    "1",
    "Number of Searches Performed",
    "127.0.0.1",
    "ArcSight",
    "0",
    "Number of Searches",
    "/Monitor/Search/Performed",
    "Logger",
    "search:100",
    1467263340016,
    "timeframe",
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    null,
    ""
  ],
  [
    "2E047-2A@Local",
    1467263340025,
    "Local",
    "Logger",
    1467263789004,
```

```
    "",
    1467263340025,
    "CurrentValue",
    "127.0.0.1",
    6950,
    1,
    1467263340025,
    "6.3.0.30273.0",
    "1",
    "Platform Memory Usage",
    "127.0.0.1",
    "ArcSight",
    "0",
    "MB Used",
    "/Monitor/Memory/Usage/Platform",
    "Logger",
    "memory:100",
    1467263340025,
    "timeframe",
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    "",
    null,
    ""
  ],
  [
    "2E047-2B@Local",
    1467263340039,
    "Local",
    "Logger",
    1467263789004,
    "",
    1467263340039,
    "SinceLastMonitorEvent",
    "127.0.0.1",
    0,
    1,
    1467263340039,
    "6.3.0.30273.0",
    "1",
    "Individual Receiver EPS",
    "127.0.0.1",
    "ArcSight",
    "0",
    "EPS",
    "/Monitor/Receiver/EPS/Individual",
    "Logger",
```

```
    "eps:102",  
    1467263340039,  
    "timeframe",  
    "UDP",  
    "up",  
    "UDP Receiver",  
    "Receiver Type",  
    "Receiver name",  
    "STATUS",  
    0,  
    "EVENT COUNT"  
  ]  
]  
}
```

Publication Status

Released: May 31, 2023

Updated: Monday, October 2, 2023

Send Documentation Feedback

If you have comments about this document, you can [contact the documentation team](#) by email. If an email client is configured on this computer, click the link above and an email window opens with the following information in the subject line:

Feedback on Web Services API Guide (Logger 7.3)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to documentation-feedback@microfocus.com.

We appreciate your feedback!