



Hewlett Packard
Enterprise

HPE Security ArcSight Logger

Software Version: 6.2

Web Services API Guide

March 31, 2016

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise 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. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

The network information used in the examples in this document (including IP addresses and hostnames) is for illustration purposes only.

HPE Security ArcSight products are highly flexible and function as you configure them. The accessibility, integrity, and confidentiality of your data is your responsibility. Implement a comprehensive security strategy and follow good security practices.

This document is confidential.

Restricted Rights Legend

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

© Copyright 2016 Hewlett Packard Enterprise Development, LP

Follow this link to see a complete statement of copyrights and acknowledgements:

<https://www.protect724.hpe.com/docs/DOC-13026>

Support

Contact Information

Phone	A list of phone numbers is available on the HPE Security ArcSight Technical Support Page: https://softwaresupport.hp.com/documents/10180/14684/esp-support-contact-list
Support Web Site	https://softwaresupport.hp.com
Protect 724 Community	https://www.protect724.hpe.com

Contents

About This Guide	7
Chapter 1: Logger's SOAP Web Services	9
Setting Up Your Development Environment	9
Accessing the SOAP Web Services	9
Setting an Authentication Token	10
Chapter 2: SOAP Login Service	11
extendSession	11
getVersion	11
login	11
logout	12
Chapter 3: SOAP Search Service	13
How the Search API Works	13
Returning Specific Fields in Search Results	14
endSearch	15
getDataforRowIds	15
getHeader	16
getNextTuples	16
hasMoreTuples	17
startSearch	17
Example: Searching for Events	17
Chapter 4: SOAP Report Service	21
getDeviceGroups	21
getDevices	21
getDevicesInDeviceGroup	22
getReportGroups	22
getSubGroups	22

getReportsInGroup	22
getStorageGroups	23
runReport	23
Example: Running a Report	24
Example: Passing Parameters when Running a Report	29
 Chapter 5: Logger's RESTful Web Services	 31
About Logger's RESTful Web Services	31
Error Messages	31
Example Search API Error Messages	31
Supported REST Web Service Clients	32
Example: Making a Login Request Using Perl Script	32
Example: Making a Login request using Curl	33
 Chapter 6: RESTful Login Service	 34
login	34
Resource URL	34
Parameters	34
Response	35
Example Login Request	35
logout	35
Resource URL	35
Parameters	36
Response	36
Example Logout Request	36
 Chapter 7: RESTful Search Service	 37
HTTP Status Codes	37
Date/Time Format	38
search	39
Resource URL	39
Parameters	39
Response	40
Example	41
status	41
Resource URL	41
Parameters	42

Response	42
Example	43
histogram	43
Resource URL	43
Parameters	43
Response	44
Example	44
drilldown	45
Resource URL	45
Parameters	45
Response	45
Example	46
events	46
Resource URL	46
Parameters	46
Response	47
Example	47
raw_events	48
Resource URL	48
Parameters	48
Response	49
Example	49
chart_data	49
Resource URL	50
Parameters	50
Response	50
Example	51
stop	51
Resource URL	51
Parameters	52
Response	52
Example	52
close	52
Resource URL	52
Parameters	53
Response	53
Example	53
Example: Running a Search	53

Send Documentation Feedback	56
-----------------------------------	----

About This Guide

This guide describes the Web services that come included with your installation of Logger 6.2. 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.

Note: SOAP Web services are not available for trial Loggers. To take advantage of this feature, you need the Enterprise version.

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 9](#) provides information applicable to all of Logger's SOAP-based Web services.
- ["SOAP Login Service" on page 11](#) 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 13](#) provides information about Logger's SOAP-based Search Service, which enables you to run search queries on Logger.
- ["SOAP Report Service" on page 21](#) provides information about Logger's SOAP-based Report Service, which enables you to run report on Logger.
- ["Logger's RESTful Web Services" on page 31](#) provides information applicable to all of Logger's RESTful Web services.
- ["RESTful Login Service" on page 34](#) 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 37](#) 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 Logger Administrator's Guide, available from the ArcSight User Community at <https://protect724.hp.com>.

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](#) 9
- [Accessing the SOAP Web Services](#) 9
- [Setting an Authentication Token](#) 10

Note: HPE is planning to move all of Logger's Web Services to simpler RESTful services. The SOAP Web Services will be deprecated in a future release. Therefore, we encourage users to move toward using the RESTful Web services where available.

SOAP Web services are not available for trial Loggers. To take advantage of this feature, you need the Enterprise version.

Setting Up Your Development Environment

Be aware of the following requirements when setting up your development environment:

- Java Web service clients require Java 8 or Java-1.7.0-openjdk-1.7.0.71-2.5.3.1.el6.x86_64.
- 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`, `core-ws-client.jar`, and `manager-ws-client.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 6.2.
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:

Note: To access the Logger API remotely, be sure change your endpoint to `<LoggerHost:Port>` instead of using `localhost:8080`, which is the default in the WSDL.

On a Logger appliance:

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

On a software Logger:

```
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 11](#). For more information about ReportService, see ["SOAP Search Service" on page 13](#). For more information about SearchService, see ["SOAP Report Service" on page 21](#).

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);
```

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	11
• getVersion	11
• login	11
• logout	12

Note: SOAP Web services are not available for trial Loggers. To take advantage of this feature, you need the Enterprise version.

extendSession

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

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

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

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 6.2, the Web services version is 1.0.0.0.2.

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);
```

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.

Chapter 3: SOAP Search Service

This section describes the API calls you can use to perform a search on Logger. It covers the following topics:

• How the Search API Works	13
• Returning Specific Fields in Search Results	14
• endSearch	15
• getDataforRowIds	15
• getHeader	16
• getNextTuples	16
• hasMoreTuples	17
• startSearch	17
• Example: Searching for Events	17

Note: SOAP Web services are not available for trial Loggers. To take advantage of this feature, you need the Enterprise version.

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.

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
- `_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 the previous page](#), 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 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, 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:

• getDeviceGroups	21
• getDevices	21
• getDevicesInDeviceGroup	22
• getReportGroups	22
• getSubGroups	22
• getReportsInGroup	22
• getStorageGroups	23
• runReport	23
• Example: Running a Report	24
• Example: Passing Parameters when Running a Report	29

Note: SOAP Web services are not available for trial Loggers. To take advantage of this feature, you need the Enterprise version.

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 **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](#) 31
- [Error Messages](#) 31
- [Supported REST Web Service Clients](#) 32

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 34](#).
- **Search Service:** For more information, see ["RESTful Search Service" on page 37](#).

Tip: RESTful services for reports are not currently supported. For reporting, use the SOAP Web service, described in ["SOAP Report Service" on page 21](#).

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 34](#).

Note: Java Web service clients require Java 8 or Java-1.7.0-openjdk-1.7.0.71-2.5.3.1.el6.x86_64.

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.

- [login](#)34
- [logout](#)35

HP recommends that you use POST instead of GET when making a call to the REST loginO 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.

Name	Type	Required	Description
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
204	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.

• HTTP Status Codes	37
• Date/Time Format	38
• search	39
• status	41
• histogram	43
• drilldown	45
• events	46
• raw_events	48
• chart_data	49
• stop	51
• close	52
• Example: Running a Search	53

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

Status Code	Description
	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	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 the previous page for the format. 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",	The list of fields (display name, not CEF) in a array to be used to calculate summary when field_summary is true.

Name	Type	Required	Default	Description
			"name"]	
field_summary	Boolean		false	Indicates to use the field summary.
local_search	Boolean		true	Indicates the search is local only, and does not include peers. Set to false if you want to include peers in the search.
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 \\.
search_type	String		interactive	The search type. Only the default value, interactive, is supported. Interactive searches send a query to the server and return the query output.
start_time	String		2 hours	<p>A string defining the beginning date and time of the search. See "Date/Time Format" on page 38 for the format.</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> <div>Note: This timeout is only two minutes. If you need to keep the search longer, increase this number.</div>

Response

This request returns the following status code or one of the status codes listed in ["HTTP Status Codes" on page 37](#).

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 31](#).

Example

This example performs a search with the query "deviceAddress CONTAINS "16.103.210.181" and the time range is from 04/02/2014 22:08:44 to 05/02/2014 22:08:44 in PDT.

```
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"
}
```

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 37](#).

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".
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".

For information about returned error messages, see ["Error Messages" on page 31](#).

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": "complete",
  "result_type": "histogram",
  "hit": 190,
  "scanned": 380,
  "elapsed": "00:00:00.322",
  "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.

Name	Type	Required	Description
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 37.

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 31](#).

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.
end_time	String	Yes	A string defining the end date and time of the search. See "Date/Time Format" on page 38 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 38 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 37](#).

For information about returned error messages, see ["Error Messages" on page 31](#).

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.

Name	Type	Required	Default	Description
offset	Number		0	The offset from the first event.

Response

This request returns one of the status codes listed in ["HTTP Status Codes" on page 37](#).

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 31](#).

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.

Name	Type	Required	Description
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 37](#).

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 31](#).

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.

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 37](#).

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 31](#).

Example

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 37.

For information about returned error messages, see "[Error Messages](#)" on page 31.

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 37.

For information about returned error messages, see "[Error Messages](#)" on page 31.

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 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/">
```

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 system, click the link above and an email window opens with the following information in the subject line:

Feedback on Web Services API Guide (Logger 6.2)

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 arc-doc@hpe.com.

We appreciate your feedback!