Riverbed Cascade Shark Common REST API v1.1

Created Jan 16, 2024 at 02:01 PM

Overview

This document describes version 1.1 of the Riverbed Common REST API as implemented by Cascade Shark systems.

The Common REST API is used to obtain general system information and for authentication.

It is assumed that the reader has practical knowledge of RESTful APIs, so the documentation does not go into detail about what REST is and how to use it. Instead the documentation focuses on what data can be accessed or modified, how to access it, and how to encode requests and responses.

The Resources section lists the supported REST resources and the methods supported on these resources. For each operation, the document describes what the operation does, the specific HTTP method and URL used, the data types used for requests and responses (if any) and any required or optional URL parameters.

The Errors section lists the various error codes that may be returned from REST API operations.

Data Encoding

Most resources exposed by the API support both XML and JSON encoding for requests and responses. The selection of the specific encoding is accomplished through the use of HTTP headers.

The Accept header should be included with all API requests, and it is used to control the encoding of the response body. To specify XML encoding, the header should be set to Accept: text/xml, and to specify JSON encoding, the header should be set to Accept: application/json. If the Accept header is omitted, the default encoding is XML.

The Content-Type header must be included with all PUT or POST requests that include a request body. To specify XML encoding, the header should be set to Content-Type: text/xml. To specify JSON encoding, the header should be set to Content-Type: application/json.

Some resources support alternative content types for requests and responses, as identified in the specific resource documentation below.

Authorization

This common API and other service-specific APIs support various methods of user authentication and authorization.

- **BASIC** (*HTTP Basic Authentication*): The username and password are passed using the Authorization HTTP header in each request.
- **COOKIE** (*Cookie-based Session Authentication*): A valid username and password combination are transmitted in an explicit login request which returns a session identifier. Subsequent requests include this session identifier as a HTTP cookie.

Resources

information: ping

Check availability of the system

```
GET https://{device}/api/common/1.1/ping
```

Authorization

This request does not require authorization.

Response Body

On success, the server does not provide any body in the responses.

information: list services

List the service identifier and version for the various API services available on this system.

```
GET https://{device}/api/common/1.1/services
```

Authorization

This request does not require authorization.

Response Body

On success, the server returns a response body with the following structure:

JSON

```
[
  {
    "id": string,
    "versions": [
      string
    ]
  }
]

Example:
[
  {
    "id": "common",
    "versions": [
      "1.0",
      "1.1"
    ]
  },
  {
    "id": "shark",
    "versions": [
      "3.2",
      "4.0",
      "5.0",
      "5.1"
    ]
  }
]
```

Property Name	Type	Description	Notes
services	*<array of <object>>*	List of common services available on this Shark	
services[service]	*<object>*	Description of an available service	
services[service].id	*<string>*	Identifier for the service	
services[service].versions	*<array of <string>>*	Available versions for service 'id'	
services[service].versions[version]	*<string>*		

information: get system information

Get basic information about the system, including version, model, and management addresses.

```
GET https://{device}/api/common/1.1/info
```

Authorization

This request does not require authorization.

Response Body

On success, the server returns a response body with the following structure:

JSON

```
{
  "sw_version": string,
  "hw_version": string,
  "device_name": string,
  "mgmt_addresses": [
    string
  ],
  "serial": string,
  "model": string
}

Example:
{
  "mgmt_addresses": [
    "172.16.222.131"
  ],
  "sw_version": "10.0.0000.0000",
  "serial": "N/A",
  "model": "vShark",
  "device_name": "my_shark"
}
```

Property Name	Type	Description	Notes
info	*<object>*	General information about this Shark	
info.sw_version	*<string>*	Software version of this Shark	
info.hw_version	*<string>*	Hardware version of this Shark (does not apply to Shark VE)	Optional
info.device_name	*<string>*	Host name for this Shark	
info.mgmt_addresses	*<array of <string>>*	Management IP addresses for this Shark	
info.mgmt_addresses[address]	*<string>*		
info.serial	*<string>*	Serial number of this Shark	
info.model	*<string>*	Model of this Shark	

authentication: login

Authenticate to the system for session-based authentication. The response will include the information needed to construct a session cookie, and will also include the Set-Cookie HTTP header.

```
POST https://{device}/api/common/1.1/login
```

Authorization

This request does not require authorization.

Request Body

Provide a request body with the following structure:

JSON

```
{
  "username": string,
  "password": string,
  "purpose": string
}

Example:
{
  "username": "user1",
  "password": "MyPassWord",
  "purpose": "Logging in to test this Shark."
}
```

Property Name	Type	Description	Notes
login	*<object>*	Login request information for a Shark session	
login.username	*<string>*	The user account to log in with	
login.password	*<string>*	The password of the given account	
login.purpose	*<string>*	The stated purpose of the login session	Optional; Should only be included if specify_purpose is enabled in the auth_info structure

Response Body

On success, the server returns a response body with the following structure:

JSON

```
{
  "session_key": string,
  "session_id": string
}

Example:
{
  "session_key": "pilot_session_id",
  "session_id": "b9d2e3b2-32b7-11e2-b4da-000c29c8cc69"
}
```

Property Name	Type	Description	Notes
login	*<object>*	Information available in response to a successful login	
*login.*session_key	*<string>*	Cookie name used to identify the session	
*login.*session_id	*<string>*	Unique session identifier ID	Should be set as the value of the 'session_key' cookie

authentication: authentication info

Get information required to authenticate to the system.

```
GET https://{device}/api/common/1.1/auth_info
```

Authorization

This request does not require authorization.

Response Body

On success, the server returns a response body with the following structure:

JSON

```
{
  "supported_methods": [
    string
  ],
  "specify_purpose": boolean,
  "login_banner": string,
  "allow_password_save": boolean
}

Example:
{
  "supported_methods": [
    "BASIC",
    "COOKIE",
    "OAUTH_2_0"
  ],
  "specify_purpose": false,
  "login_banner": "",
  "allow_password_save": false
}
```

Property Name	Type	Description	Notes
auth_info	*<object>*	Information about authentication protocols	
*auth_info.*supported_methods	*<array of <string>>*	Available authentication methods	
*auth_info.*supported_methods[method]	*<string>*	Authentication method	Values: BASIC, COOKIE, OAUTH_2_0
*auth_info.*specify_purpose	*<boolean>*	Indication if the user should be prompted to include a purpose with the login request	
*auth_info.*login_banner	*<string>*	Banner to be displayed on login page	
*auth_info.*allow_password_save	*<boolean>*	Indicates if the user is allowed to store the client password locally	Optional

authentication: logout

Log out from the system. The request must include a session cookie and will invalidate that cookie for future requests.

```
POST https://{device}/api/common/1.1/logout
```

Authorization

This request requires authorization.

Request Body

Do not provide a request body.

Response Body

On success, the server does not provide any body in the responses.

Error Codes

In the event that an error occurs while processing a request, the server will respond with appropriate HTTP status code and additional information in the response body:

```
{
    "error_id":   "{error identifier}",
    "error_text": "{error description}",
    "error_info": {error specific data structure, optional}
}
```

The table below lists the possible errors and the associated HTTP status codes that may returned.

Error ID	HTTP Status	Comments
REQUEST_INVALID_INPUT	400	The request is invalid
AUTH_REQUIRED	401	Missing authentication credentials
AUTH_INVALID_CREDENTIALS	401	Invalid user name or password
AUTH_INVALID_SESSION	401	The authentication session has timed out or is invalid
AUTH_EXPIRED_PASSWORD	401	Account password has expired
AUTH_INVALID_CODE	401	The Oauth access code is invalid
AUTH_EXPIRED_TOKEN	401	The Oauth token has expired
AUTH_EXPIRED_CODE	401	The Oauth access code has expired
AUTH_DISABLED_ACCOUNT	403	Account has been disabled
AUTH_FORBIDDEN	403	Account does not have privileges for this request
AUTH_INVALID_TOKEN	403	The Oauth token is invalid
RESOURCE_NOT_FOUND	404	The requested resource was not found
HTTP_INVALID_METHOD	405	The requested method is not supported by this resouce
INTERNAL_ERROR	500	Internal error occurred