Overview

This document describes version 1.0 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/1.0/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/1.0/services
```
Authorization
This request does not require authorization.

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

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

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>services</td>
<td>&lt;array of &lt;object&gt;&gt;</td>
<td>List of common services available on this Shark</td>
<td></td>
</tr>
<tr>
<td>services[service]</td>
<td>&lt;object&gt;</td>
<td>Description of an available service</td>
<td></td>
</tr>
<tr>
<td>services[service].id</td>
<td>&lt;string&gt;</td>
<td>Identifier for the service</td>
<td></td>
</tr>
<tr>
<td>services[service].versions</td>
<td>&lt;array of &lt;string&gt;&gt;</td>
<td>Available versions for service 'id'</td>
<td></td>
</tr>
<tr>
<td>services[service].versions[version]</td>
<td>&lt;string&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**information: get system information**

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

GET https://{device}/api/common/1.0/info

Authorization
This request does not require authorization.

Response Body
On success, the server returns a response body with the following structure:
Example:
{
"mgmt_addresses": ["172.16.222.131"],
"sw_version": "10.0.0000.0000",
"serial": "N/A",
"model": "vShark",
"device_name": "my_shark"
}

Example:
{
"username": "user1",
"password": "MyPassWord",
"purpose": "Logging in to test 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/1.0/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."
}

### JSON

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>&lt;object&gt;</td>
<td>Login request information for a Shark session</td>
<td></td>
</tr>
<tr>
<td>login.username</td>
<td>&lt;string&gt;</td>
<td>The user account to log in with</td>
<td></td>
</tr>
<tr>
<td>login.password</td>
<td>&lt;string&gt;</td>
<td>The password of the given account</td>
<td></td>
</tr>
<tr>
<td>login.purpose</td>
<td>&lt;string&gt;</td>
<td>The stated purpose of the login session</td>
<td>Optional: Should only be included if specify_purpose is enabled in the auth_info structure</td>
</tr>
</tbody>
</table>
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"
}
```

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>&lt;object&gt;</td>
<td>Information available in response to a successful login</td>
<td></td>
</tr>
<tr>
<td>login.session_key</td>
<td>&lt;string&gt;</td>
<td>Cookie name used to identify the session</td>
<td></td>
</tr>
<tr>
<td>login.session_id</td>
<td>&lt;string&gt;</td>
<td>Unique session identifier ID</td>
<td>Should be set as the value of the 'session_key' cookie</td>
</tr>
</tbody>
</table>

authentication: authentication info

Get information required to authenticate to the system.

**GET** https://{device}/api/common/1.0/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
}
```

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

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Type</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth_info</td>
<td>&lt;object&gt;</td>
<td>Information about authentication protocols</td>
<td></td>
</tr>
<tr>
<td>auth_info.supported_methods</td>
<td>&lt;array of &lt;string&gt;&gt;</td>
<td>Available authentication methods</td>
<td></td>
</tr>
<tr>
<td>auth_info.supported_methods[method]</td>
<td>&lt;string&gt;</td>
<td>Authentication method</td>
<td>Values: BASIC, COOKIE, OAUTH_2_0</td>
</tr>
<tr>
<td>auth_info.specify_purpose</td>
<td>&lt;boolean&gt;</td>
<td>Indication if the user should be prompted to include a purpose with the login request</td>
<td></td>
</tr>
<tr>
<td>auth_info.login_banner</td>
<td>&lt;string&gt;</td>
<td>Banner to be displayed on login page</td>
<td></td>
</tr>
</tbody>
</table>

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/1.0/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:

```json
{
   "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.

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