Boxilla Northbound REST API V1.9.10 Release for External Manager

Document Number: ENG-007-100

Author: Shengyang Chen

Status: Draft

Rev: 1.9.11

Date: 26/11/2025

Master Document Revision History

Date	Version	Description	Author / Editor
Jan. 6th, 2020	0.1	Initial draft derived from the original northbound REST API document ENG-0007-078, with the following updates: • Added description of request parameter usage in JSON format for each API • Removed all future features including templated-related and connection-group-related operations • Removed all descriptions of appliance southbound REST API (user logout, launch/break active connections) • Updated context based on the consolidated review comments	Shengyang Chen
Jan. 17th, 2020	0.2	Update context based on the second group of consolidated review comments.	Shengyang Chen
Jan. 17th, 2020	1.0	Moving to approved.	Shengyang Chen
Feb.17th, 2020	1.1	Parameter consistency update: • replacing character "-" with "_" in API request • updating valid values for several parameters • ensuring consistency over various API types (GET/POST/PUT/DELETE) Error case and response context update.	Shengyang Chen

Mar.27th, 2020	1.2	Added connection property compatibility explanation for usb_rediretion of viaTx connections and host for VMPool connections. Error case and response context update. Added description of unique identifier features for GET APIs for device status and connections. Removed description related to SSL certificate installation.	Shengyang Chen
May.5th, 2020	1.3	Added definition of GET API for active connection summary. URL correction of the API for editing connection properties. Response message update for user login API for active directory and non-active-directory users.	Shengyang Chen
Jun.19th, 2020	1.4	Added definitions of zone-related APIs Added compression mode support parameters in connection APIs Added APIs for user favorites Updated REST API version from v1.0 to v1.1, representing support for zone and compression mode	Shengyang Chen
Jun.2nd, 2021	1.5	Added appendix section of recommended northbound REST API operation work-flow sequence	Shengyang Chen
Mar. 7, 2022	1.6	Added definitions for Remote App-oriented operations. Updated Northbound REST API URI Format, added new root element into API hierarchy definition. Updated POST APIs, added upload private key API.	Julian Mamokin
Mar. 30, 2022	1.7	Added definitions for Time-oriented operations. Updated Northbound REST API URI Format, added new root element into API hierarchy definition.	Julian Mamokin

		Updated GET APIs, added time API.	
Nov. 28, 2022	1.8	Added APIs for retrieving Boxilla backup files	Shilei Mao
Feb. 17, 2023	1.9.0	Extended Terminate KVM Active Connection and KVM User Logout APIs to include new optional parameter “splash_screen”: “Yes” | “No”.	Aidan Quinn.
September 11th, 2023	1.9.1	Renamed the TX settings parameter “DVI Optimised” to “DVI/DP Optimised” as to align with Boxilla GUI.	Shilei Mao
Oct. 20th 2023	1.9.2	Added new APIs for getting preset details and launching preset.	Shilei Mao
Nov. 17th 2023	1.9.3	Updated preset APIs to introduce username for preset list and details requests.	Shilei Mao
Dec. 18th 2023	1.9.4	Added “view_only” property for active connection summary response message.	Shilei Mao
Feb. 20th, 2024	1.9.5	Added error response for TXPair connections for preset launch. Added error response for KVM connection creation, updates and launches.	Shilei Mao
Mar. 14th, 2024	1.9.6	Added descriptions for exclusive connections: Create/update/launch connection. Updated error response for launch connection preset with exclusive mode changes.	Shilei Mao
Apr. 05th, 2024	1.9.7	Updated bxa-api/users/kvm API to include new field Description.	Shilei Mao
Apr. 16th, 2024	1.9.8	Updated bxa-api/users/kvm/login API to make the rx_list and forced parameters optional.	Shilei Mao
May. 27th, 2024	1.9.9	Updated the response message for TXPair connections on active connection status and active connection summary API.	Shilei Mao
September 26th, 2024	1.9.10	Updated the NBR definition for getting preset list.	Shilei Mao
May 19th, 2025	1.9.11	Included new NBR definitions for video wall support.	Shilei Mao

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

Glossary of REST-related Terms

Term	Description
REST	Representational State Transfer. An architectural style that uses HTTP as the transport protocol for the message requests and responses.
API	Application Programming Interface. Enables different systems to interact with each other programmatically.
URI	Uniform Resource Identifier. A string containing an identifier (usually a name or an address) which refers to a resource in the web.
GET	A REST operation to retrieve a resource.
POST	A REST operation to create a resource.
PUT	A REST operation to update or create within an existing resource.
DELETE	A REST operation to remove a resource.
JSON	JavaScript Object Notation. A lightweight syntax containing objects and arrays, usually used (instead of XML) to return information from a REST API.

1 Introduction

Blackbox hardware devices are managed on Boxilla via web services that provide functionalities of creating, removing, updating and validating KVM devices, connections and user profiles. The corresponding functionalities are required as independent components which can be integrated into and accessed potentially by third-party management platforms used by customers for Blackbox device management, other than accessing devices directly from Boxilla.

This document provides the definition of Boxilla northbound REST API that acts as the management interface integrated with any third-party management tools. Requirements on the REST API functionalities are listed but not limited as follows:

- Add support to create/edit/delete profiles of KVM users and get information of individual/multiple users
- Add support to log KVM users in to or out from KVM receivers
- Add support to update API authentication credentials (username/password)
- Add support to get list of transmitters/receivers, properties and status
- Add support to reboot individual/multiple devices
- Add support to create/delete KVM connections, get connection information and edit connection properties
- Add support to retrieve Boxilla backup files
- Add support to launch connection preset
- Add support to launch video wall and check status

The northbound REST API is designed as one of the new features for Boxilla 3.5. It is NOT supported by Boxilla with version older than 3.5.

2 Boxilla REST API for External Management Platform Integration

2.1 General Procedures

Version 1.1 of the northbound REST APIs listed in this document provides **administrator** level of operations to Boxilla system and the associated resources. For this reason, they are **disabled** and cannot be accessed by external management platforms by default. Configuration of enabling northbound REST APIs is managed by **Boxilla via its web interface**. Potential REST API users should enable the northbound REST APIs (and only enable them when required) before using them, otherwise all requests via the APIs would fail and with no response.

2.2 Northbound REST API URI Format

The URI of each northbound REST API designed in this document is formatted as follows:

scheme "://" authority "/" path ["?" query] ["#" fragment]

The path field within the format is defined using the following hierarchy rules:

- users
    - kvm
    - system
- devices
    - kvm
    - switches
    - dkm
- peripherals
    - usb-extenders
- connections
    - kvm
    - dkm
    - presets
- zones
- remoteapp
- time
- backups

- videowall
  - list
  - activate
  - terminate
  - active
    - status

As the northbound REST API is designed for the majority of REST functionalities for Boxilla in general, the authority field within the URI is defined as **bxa-api**.

The following sections list the URLs of all the northbound REST APIs accessed using the GET, POST, PUT and DELETE verbs, respectively.

2.2.1 GET APIs

Scope	URL	API version	
		v1	v1.1
Users	bxa-api/users/kvm	✓	✓
Devices	bxa-api/devices/kvm	✓	✓
	bxa-api/devices/kvm/properties	✓	✓
	bxa-api/devices/kvm/status	✓	✓
Connections	bxa-api/connections/kvm	✓	✓
	bxa-api/connections/kvm/active	✓	✓
	bxa-api/connections/kvm/active/summary	✓	✓
	bxa-api/connections/presets		✓
	bxa-api/connections/presets/details		✓
	bxa-api/connections/presets/active/summary		✓
Zones	bxa-api/zones		✓
	bxa-api/zones/all		✓
Versioning	bxa-api/version	✓	✓
Time	bxa-api/time		✓
Backups	bxa-api/backups/list		✓
	bxa-api/backups/download		✓
Video walls	bxa-api/videowall/list		✓
	bxa-api/videowall/active/status		✓

2.2.2 POST APIs

Scope	URL	API version	
		v1	v1.1
Users	bxa-api/users/kvm	✓	✓

	bxa-api/users/kvm/login	✓	✓
	bxa-api/users/kvm/logout	✓	✓
	bxa-api/users/kvm/favs		✓
Devices	bxa-api/devices/kvm/reboot	✓	✓
	bxa-api/devices/kvm/reboot-all	✓	✓
	bxa-api/devices/kvm/rx/zones		✓
Connections	bxa-api/connections/kvm	✓	✓
	bxa-api/connections/kvm/active	✓	✓
	bxa-api/connections/presets/activate		✓
Zones	bxa-api/zones		✓
	bxa-api/zones/devices/kvm/rx		✓
	bxa-api/zones/connections/kvm		✓
Remote App	bxa-api/remoteapp/cli_data_key		✓
Video walls	bxa-api/videowall/activate		✓

2.2.3 PUT APIs

Scope	URL	API version	
		v1	v1.1
Users	bxa-api/users/kvm	✓	✓
	bxa-api/users/kvm/connections	✓	✓
Devices	bxa-api/devices/kvm/properties/rx	✓	✓
	bxa-api/devices/kvm/properties/tx	✓	✓
Connections	bxa-api/connections/kvm	✓	✓
Zones	bxa-api/zones		✓
REST Authentication	bxa-api/users/rest/username	✓	✓
	bxa-api/users/rest/password	✓	✓
Video walls	bxa-api/videowall/terminate		✓

2.2.4 DELETE APIs

Scope	URL	API version	
		v1	v1.1
Users	bxa-api/users/kvm	✓	✓
	bxa-api/users/kvm/all	✓	✓
Connections	bxa-api/connections/kvm	✓	✓
	bxa-api/connections/kvm/all	✓	✓
	bxa-api/connections/kvm/active	✓	✓
Zones	bxa-api/zones		✓
	bxa-api/zones/all		✓

2.3 REST API Header

The header of each REST API defined in this document contains universal information that applies to all APIs, validated by Boxilla. A standard example of the header of a Boxilla northbound REST API in this document is defined as follows:

```
Accept: application/json

Content-Type: application/json

Authorization: Basic bG9sOnNlY3VyZQ==

Accept-version: v1.1
```

Each field of the header is discussed in the following sections.

2.3.1 Request/Response Formatting

The "`Accept: application/json`" and "`Content-Type: application/json`" fields in each REST API request header indicate that the body of API request and response is in JSON format.

If these fields are not provided in the API request header, a standard HTTP-302 response would be returned by Boxilla.

2.3.2 Authentication/Authorization and Security

The Northbound REST APIs listed in this document use HTTPs by default, configurable on port 443.

The "`Authorization: Basic ......`" field in the header of each API request indicates that all requests/responses of the REST APIs in this document use basic authentication method for authentication/authorization. A universal REST API user with a username-password credential pair is designed for operations over REST APIs for all KVM devices and connections, independent from the associated KVM user accounts. This credential pair is deployed in Boxilla with uniqueness, encoded with Base64 algorithm and included after the "Basic" context in any API request header.

The default REST API authentication/authorization credential pair used in Boxilla is as follows:

- Username: **REST_BbAdminUser**
- Password: **Boxill@2020**

As mentioned in section 2.1, the northbound REST APIs are with administrator level access of Boxilla system and resources, which is guaranteed by using this default REST credential pair. New credential pairs potentially added in future design can be used to change the access level.

2.3.3 API Versioning

Versioning is handled in the customized REST API request/response header to preserve URIs between versions using the "`Accept-version`" field.

The default version number to be used is ***v1.1***. REST APIs defined in the previous versions (v1) are still supported within this version.

2.4 Error Response Cases for all APIs

The success/error response of each northbound REST API in this document follows the HTTP-status-code style of response, using status code to represent various cases together with details of the response. The API response header includes the version of API being used via the request and the response body contains the detailed description of the response status in a "message" field in JSON.

The following error response cases apply to all the REST APIs defined in this document:

2.4.1 Response - Error: Bad Request

This error response indicates any non-specific error that might be caused by internal errors within the API, with the following format:

```
HTTP/1.1 400 Bad Request
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "400",
  "message": "[Error details]."
}
```

The detailed error message varies depending on the actual error case related. The individual error case would be discussed in the definition of each REST API, respectively.

2.4.2 Response - Error: UnAuthorized

This error response indicates that the operation related to the API request is not authorized on Boxilla, usually caused by an invalid authentication credential.

```
HTTP/1.1 401 UnAuthorized
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": “401”,
  "message": "Operation is not authorized."
}
```

2.4.3 Response - Error: Not Found

This error response indicates that the API request URL does not exist. The error case is handled with a HTML-format error response. Version 1 of northbound REST API does not customize this error response in JSON.

```
HTTP/1.1 404 Not Found
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Action Controller: Exception caught</title>
</head>
<body>
<center><br><br><img src="/assets/bbb_logo.png"><br><br>
<h2>404: Resource not found.</h2><br><br>
<h3>Please return and try again</h3>
<a href="/"><button class="btn btn-default" type="button">Dashboard</button></a><br><br>
<small><i>Routing Error</i></small>
</body>
</html>
```

2.5 APIs for REST User Related Operations

The APIs for REST user management include the following functionalities:

- Change username/password of the REST user account credential

The associated definition of each API is listed as follows:

2.5.1 API for Editing REST Username

2.5.1.1 Request - Uri

Uri	bxa-api/users/rest/username
Method	PUT

2.5.1.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Current username credential of the REST API user. Maximum length 64.
	new_username	String	New username credential of the REST API user to change to. Maximum length 64.

2.5.1.3 Request Body JSON Format

```
{
  "username": "[username]",
  "new_username": "[new_username]"
}
```

2.5.1.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "Successfully updated REST username."
}
```

2.5.1.5 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- The username credential provided in the parameters does not match the existing valid credential. The associated error message in the response body is as follows:
  "**message**": "REST username *[username]* does not exist."

2.5.1.6 Response – Error: REST Username Credential already Exists

This error indicates that the new REST username credential to be updated to (specified by the "new_username" parameter) already exists.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Cannot update REST username to [new_username], [new_username] already
exists."
}
```

2.5.1.7 Response – Error: Northbound REST API is not enabled for the REST User

This error indicates that northbound REST API is disabled for the specified REST user specified by the "username" parameter.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Editing REST authentication credentials is forbidden for user [username]. REST API is
not accessible for this user."
}
```

2.5.2 API for Editing REST Password

2.5.2.1 Request - Uri

Uri	bxa-api/users/rest/password
Method	PUT

2.5.2.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Current username credential of the REST API user. Maximum length 64.
	new_password	String	New password credential of the REST API user to change to. Maximum length 32.

2.5.2.3 Request Body JSON Format

```
{
  "username": "[username]",
  "new_password": "[new_password]"
}
```

2.5.2.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Successfully updated REST user password."
}
```

2.5.2.5 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- The username credential provided in the parameters does not match the existing valid credential. The associated error message in the response body is as follows:
  "**message**": "REST username *[username]* does not exist."

2.5.2.6 Response – Error: Northbound REST API is not enabled for the REST User

This error indicates that northbound REST API is disabled for the specified REST user specified by the "username" parameter.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Editing REST authentication credentials is forbidden for user [username]. REST API is not accessible for this user."
}
```

2.6 APIs for Versioning

This API is to get versioning information related to Boxilla components, including information of the following items each with fixed format:

- Boxilla software version
  - Format is *[major_version].[minor_version].[patch].[build_number]*, e.g. 3.4.1.5120
- Northbound REST API version: fixed as "v1" for API version 1
- Boxilla model number: fixed as "BXAMGR"
- Boxilla serial number: e.g. AEBB19B00022

2.6.1 Request - Uri

Uri	bxa-api/version
Method	GET

2.6.2 Request – Params

This API does not require any parameters within the request body.

2.6.3 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": “200”,
  "message": {
    "software-version": “[boxilla software version]”,
    "api_version": “v1”,
    "model_number": “BXAMGR”,
    "serial_number": “[boxilla serial number]”
  }
}
```

2.7 APIs for Time-oriented Operations

This API is to retrieve current Boxilla time, including UTC time and Local time. All values are returned in **epoch** format.

- local_time – Boxilla local time (timezone aware)
- utc_time – Boxilla UTC time

2.7.1 Request - Uri

Uri	bxa-api/time
Method	GET

2.7.2 Request – Params

This API does not require any parameters within the request body.

2.7.3 Response - Success

```
HTTP/1.1 200 OK

Content-Type: application/json;
```

```
Accept-version: v1.1
Body:
{
  "code": "200",
  "message": {
    "local_time": "1648628865",
    "utc_time": "1648628865"
  }
}
```

2.8 APIs for User-oriented Operations

The APIs for Boxilla user operations include functionalities as follows:

- Creating new KVM users
- Editing existing KVM user profiles
- Deleting existing KVM users
- Getting KVM users (individual/all)
- Editing associations between KVM users and connections
- Logging users in to KVM receivers
- Logging users out from KVM receivers

The associated definition of each API is listed as follows:

2.8.1 API for Creating KVM Users

This API is to create a new KVM user in Boxilla using a valid credential set (username + password) from the API input parameters, together with the property set associated to the user. For each API request, one of the following categories of user property sets is defined with the relevant parameters passed in apart from the user credential (username/password):

- Unique property: user is created with a unique set of properties defined with the API request parameters. In this case the request of the API should contain the following parameters for the user property:
  - *privilege*: identifying the privilege of the user, valid values including “Administrator”, “PowerUser” or “GeneralUser”.

- *auto_connect*: identifying whether the auto-connection option of the user property is enabled, valid values including “Yes” or “No”.
- *auto_connect_name*: identifying the name of the auto-connection if the property is enabled.
- *remote_access*: identifying whether access of the user profile via the remote windows application is enabled, valid values include “Yes” or “No”.

Version 1 of northbound REST API does not support user-related operations with non-unique properties, including template-based and system properties.

The “Requirement” column in the parameter table represents whether each parameter is mandatory, conditional-mandatory or optional, depending on specific values of some parameters as follows:

- Mandatory: the parameter is required
- Conditionally Optional: the parameter is optional depending on values of one or more other parameters.

2.8.1.1 Request - Uri

Uri	bxa-api/users/kvm
Method	POST

2.8.1.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Username credential of KVM user. Maximum length 64.
	password	String	Password credential of KVM user. Maximum length 32.
	privilege	String	User privilege. Valid values are: “Administrator” | “PowerUser” | “User”. (Case sensitive)
	remote_access	String	Enabled/disabled option for remote app access. Valid values are “Yes” or “No”. (Case sensitive)
	auto_connect	String	Enabled/disabled option for auto-connection. Valid values are “Yes” or “No”. (Case sensitive)
Conditionally Optional	auto_connect_name	String	Name of the auto-connection if enabled. Maximum length 64. **Applicable when “auto_connect” is “Yes”.**
Optional	description	String	A custom description text for user, maximum 32 characters.

2.8.1.3 Request Body JSON Format

```
{
  "username": "[username]",
  "password": "[password]",
  "privilege": "Administrator" | "PowerUser" | "User",
  "remote_access": "Yes" | "No",
  "auto_connect": "Yes" | "No",
  "auto_connect_name": "[auto_connect_name]",  // only add it when "auto_connect" is "Yes"
}
```

2.8.1.4 Response - Success

```
HTTP/1.1 201 Created

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "201",
  "message": "Created user [username]."
}
```

2.8.1.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Username/password or other required property parameters are passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- When "auto_connect" is enabled, the provided "auto_connect_name" parameter value is not the name of any existing KVM connections. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter: auto_connect_name *[auto_connect_name]* does not point to any existing KVM connections."
- "auto_connect_name" parameter is provided when "auto_connect" is disabled. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter: auto_connect_name *[auto_connect_name]* cannot be set when auto_connect is disabled. "

2.8.1.6 Response - Error: License-related Failure

This error response indicates that the Boxilla license installed on the server is invalid, expired or limitation of the maximum number of users is reached.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "License limit reached."
}
```

2.8.1.7 Response - Error: User Already Exists

This error response indicates that a KVM user record with the specified username already exists and the API request is attempting to create a duplicate record.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Duplicate name received."
}
```

2.8.2 API for Editing KVM Users

This API is to edit the profile/properties of an existing KVM user. The request of the API should contain the user profile/property parameters to be updated.

2.8.2.1 Request - Uri

Uri	bxa-api/users/kvm
Method	PUT

2.8.2.2 *Request - Params*

Requirement	Name	Type	Description
Mandatory	username	String	Username credential of KVM user. Maximum length 64.
	password	String	Password credential of KVM user. Maximum length 32.
	privilege	String	User privilege. Valid values are: “Administrator” | “PowerUser” | “User”. (Case sensitive)
	remote_access	String	Enabled/disabled option for remote app access. Valid values are “Yes” or “No”. (Case sensitive)
	auto_connect	String	Enabled/disabled option for auto-connection. Valid values are “Yes” or “No”. (Case sensitive)
Optional	new_username	String	New username to be edited. Maximum length 64. Username is updated if provided.
	description	String	A custom description text for user, maximum 32 characters.
Conditionally Optional	auto_connect_name	String	Name of the auto-connection if enabled. Maximum length 64.

2.8.2.3 *Request Body JSON Format*

```
{
  "username": “[username]”,
  "password": “[password]”,
  "privilege": “Administrator” | “PowerUser” | “User”,
  "remote_access": “Yes” | “No”,
  "auto_connect": “Yes” | “No”,
  "auto_connect_name": “[auto_connect_name]” // only add it when “auto_connect” is “Yes”
  "new_username": “[new_username]”   // Optional, username is updated if provided
}
```

2.8.2.4 *Response - Success*

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "Updated profile/properties for user [username]."
}
```

2.8.2.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required parameters are passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- One or more KVM users within the username list parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist."
- When “auto_connect” is enabled, the provided “auto_connect_name” parameter value is not the name of any existing KVM connections. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter: auto_connect_name *[auto_connect_name]* does not point to any existing KVM connections."
- “auto_connect_name” parameter is provided when “auto_connect” is disabled. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter: auto_connect_name *[auto_connect_name]* cannot be set when auto_connect is disabled. "

2.8.2.6 Response – Error: Changing Username to Name of Existing User

This error response indicates that the new username to be changed to belongs to an existing user. The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1

Body:
{
  "code": "403",
  "message": "Cannot update username to [new_username]. User [new_username] already
exists."
}
```

2.8.2.7 Response – Error: Attempting to Edit Properties of non-Unique Users

This error response indicates that the specified user is with non-unique properties. Editing properties of the user would be prohibited by the northbound REST API and a HTTP 403 error response is returned. The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": " Editing properties for non-unique user [username] is not supported."
}
```

2.8.3 API for Deleting All KVM Users

This API is to delete all the existing KVM users with unique properties in Boxilla.

2.8.3.1 Request - Uri

Uri	bxa-api/users/kvm/all
Method	DELETE

2.8.3.2 Request – Params

This API does not require any parameters within the request body. The version of API is included in the request header, as described in section 2.3.3.

2.8.3.3 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Successfully deleted all users."
}
```

2.8.3.4 *Response – Error: User Logged on Receivers or in Active Connection*

This error response indicates that one or more users are actively logged in receivers or are in active connections. The associated error message in the response body is as follows:

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "409",
  "message": "Fail to delete user(s) [list of active users] because one or more users are logged on receivers or in active connections."
}
```

2.8.4 API for Deleting Specific KVM Users

This API is to delete one or more KVM users with unique properties in Boxilla, specified by the request parameter which contains a list of existing usernames to be deleted.

2.8.4.1 *Request - Uri*

Uri	bxa-api/users/kvm
Method	DELETE

2.8.4.2 *Request - Params*

Requirement	Name	Type	Description
Mandatory	usernames	String array	List of names of the KVM users to be deleted. If array is empty with no usernames inside, the API request is successful but no user would be deleted.

2.8.4.3 *Request Body JSON Format*

```
{
  "usernames": ["[username_1]", "[username_2]", …]
}
```

2.8.4.4 *Response - Success*

```
HTTP/1.1 200 OK
Accept: application/json;
```

```
Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "Successfully deleted user(s) [usernames]."
}
```

2.8.4.5 Response – Success with Empty Usernames Parameter

```
HTTP/1.1 204 No Content

Accept: application/json;

Content-Type: application/json;

Accept-version: v1
```

2.8.4.6 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required parameters are passed in as empty (null parameter, not parameter as an empty array) or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- "usernames" parameter is not an array (for example, an integer is given instead) or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter usernames: *[usernames]*, expecting a String array."
- One or more KVM users within the username list parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following user(s) do(es) not exist: *[List of usernames do not exist]*"

2.8.4.7 Response – Error: Attempting to Delete Users with non-Unique Properties

This error response indicates that one or more KVM users specified in the parameters are with non-unique properties. Deleting these users would be prohibited by the northbound REST API and a HTTP 403 error response is returned. Meanwhile all the other valid users specified in the parameters would still be successfully deleted.

The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "The following users are with non-unique properties: [List of usernames]."
}
```

2.8.4.8 Response – Error: User Logged on Receivers or in Active Connection

This error response indicates that one or more users are actively logged in receivers or are in active connections. The associated error message in the response body is as follows:

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "409",
  "message": "Fail to delete user(s) [list of active users] because one or more users are logged on
receivers or in active connections."
}
```

2.8.5 API for Getting KVM Users

This API is to obtain one or more KVM users on Boxilla, specified by the request parameter.

2.8.5.1 Request - Uri

Uri	bxa-api/users/kvm
Method	GET

2.8.5.2 Request – Params

Requirement	Name	Type	Description
Optional	usernames	String array	List of names of the KVM users to get.

			If array is empty with no usernames inside, the API request is successful but no user would be returned in response body.

Note:

If no parameter is provided by the API request (parameter is null, not an empty array), the operation would be considered as getting all KVM users and the response is returned accordingly.

If the "usernames" parameter is an empty array, the API request is successful but no user is returned in the response (the "users" JSON array in response body is empty).

2.8.5.3 Request Body JSON Format

```
{
  "usernames": ["[username_1]", "[username_2]", …]  // Optional, can be null or empty array
}
```

2.8.5.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
   "users": [
    {
     "username": "[username]",
     "privilege": "Administrator" | "PowerUser" | "User",
     "auto_connect": "Yes" | "No",
     "auto_connect_name": "[auto_connect_name]",  // available if "auto_connect" is "Yes"
     "remote_access": "Yes" | "No",
     "connections": [
      {
       "connection_name": "[connection_name]"
      },
      {….},
      …. // if there are more connections
     ],
      "description": "[description text]"
     },
     {….},   // if there are more users
     ….
   ]
  }
}
```

In the “message”->”users” JSON array field in the response body, the following keys are mandatory (always be present) in each JSON element of the array:

- username
- privilege
- remote_access
- auto_connect

The “auto_connection_name” key is conditionally optional based on the value of the “auto_connect” parameter. Meanwhile the value “connections” key depends on the associated connection list of the specified user.

2.8.5.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"
- “usernames” parameter is not an array (for example, an integer is given instead) or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter usernames: *[usernames]*, expecting a String array."
- One or more KVM users within the username list parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following user(s) do(es) not exist: *[List of usernames do not exist]*"

2.8.6 API for Editing Associations between KVM Users and Connections

This API is to edit the association between a KVM user and a list of KVM connections, each of which is either attached to or detached from the KVM user.

2.8.6.1 Request - Uri

Uri	bxa-api/users/kvm/connections
Method	PUT

2.8.6.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Name of the KVM user that owns the list of KVM connections.
	connection_names	String array	List of names of the KVM connections to be associated to the KVM user. Can be an empty array.

The operation of this API depends on the value of the **connection_names** parameter as follows:

- If the parameter includes one or more valid connection names, all existing associations of KVM connections to the specified KVM user are removed and a new user-connection association is created for each KVM connection included in the parameter to the KVM user.
- If the parameter is an empty array, all existing associations of KVM connections to the specified KVM user are removed.

2.8.6.3 Request Body JSON Format

```
{
  "username": "[username]",
  "connection_names": ["[connection_1]", "[connection_2]", …]  //Mandatory, can be empty array
}
```

2.8.6.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Successfully updated connection associations for user [username]."
}
```

2.8.6.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- "connection_names" parameter value is not an array or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter connection_names: *[connection_names]*, expecting a String array."
- No KVM user exists with the specified username parameter in the request body. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist."
- No KVM connections exist with the specified connection name parameters in the request body. The associated error message in the response body is as follows:
  "**message**": "The following connection(s) do(es) not exist: *[connection_names]*."
- The maximum limit of connections (22500) can be associated to a user has been reached. The associated error message in the response body is as follows:
  "**message**": "User connection limit reached."

2.8.7 API for KVM User Login to Receivers

This API is to login a KVM user to a specific receiver or group of receivers, with a force-login option enabled or disabled. Enabling force-login would log out the alternative KVM user logged in currently, while disabling force-login results in a failure of the API request when another KVM user is logged in. The KVM user can be an active directory user whose record is not managed on Boxilla.

The force-login option is set as a parameter within the request body. Valid parameters are listed in the parameter table below.

2.8.7.1 Request - Uri

Uri	bxa-api/users/kvm/login
Method	POST

2.8.7.2 Request - Params

Requirement	Name	Type	Description

Mandatory	username	String	Username of the KVM user for logging in.
	password	String	Password of the KVM user for logging in.
Conditionally Optional	rx_list	String array	The list of receiver names that the user logs in. When it's not provided, the API is simply verify the KVM username and password.
	forced	String	String parameter indicates login operation is a force-login or not. Valid values are "Yes" or "No". **Note: force-login would break existing active connections. When the rx_list parameter is not provided, this parameter is optional and will be ignored by the API.**

2.8.7.3 Request Body JSON Format

```
{
  "username": "[username]",
  "password": "[password]",
  "rx_list": ["[receiver_1]", "[receiver_2]", ….], // Conditionally optional, API verifies the username password if this parameter not provided
  "forced": "Yes" | "No" // Conditionally optional, when the rx_list not provided, this parameter is ignored.
}
```

2.8.7.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "User [username] has successfully logged in to the target receivers."
}
```

2.8.7.5 Response - Error: Bad Request

This error response may be caused by:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"

- One or more receivers included in the rx-list parameter are offline. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) to log in is(are) offline: *[List of offline receiver names]*"
- The login functions on one or more receivers included in the rx-list parameter are malfunctional. The associated error message in the response body is as follows:
  "**message**": " "User login failed on the following receiver(s): *[List of malfunctional receiver names]*"
- The KVM user specified by the username parameter does not exist. This happens when the provided user is neither managed by Boxilla, nor active directory option is enabled. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist. "

2.8.7.6 Response - Error: UnAuthorized

This error response may be caused by:

- The REST user credentials included in the request header are invalid, as described in section 2.4.2.
- The user is a local user created on Boxilla and the login password provided in request parameters is incorrect. The associated error message in the response body is as follows:
  "**message**": "Operation is not authorized due to invalid login credentials."
- The user is an active directory user managed by active directory server and the login credentials for it are incorrect. The associated error message in the response body is as follows:
  "**message**": " Operation is not authorized. Active directory user authentication failed on one or more target receivers. Please check the active directory server configuration on your Boxilla and make sure the credentials are correct."

2.8.7.7 Response - Error: Logging in with Force-Login Disabled while Alternative User is Logged in

This error response indicates that another user is logged in to one or more receivers listed in the rx-list parameter, with the force-login option disabled. In this case the non-force-login operation from the API request would be rejected.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
```

```
Accept-version: v1

Body:

{
  "code": "403",
  "message": "Another user is logged in target receiver(s) [rx_list with user logged in]."
}
```

2.8.8 API for KVM User Logout from Specific Receivers

This API is to log out a KVM user from a specific receiver or group of receivers. Valid parameters are listed in the parameter table below.

2.8.8.1 Request - Uri

Uri	bxa-api/users/kvm/logout
Method	POST

2.8.8.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Username of the KVM user for logging out.
	rx_list	String array	List of receiver names that the user logs out.
	forced	String	An Integer number indicating that the login operation is a force-logout or not. Valid values are “Yes” or “No”. **Note: force-logout would break existing active connections**
Optional	splash_screen	String	String indicating if the splash screen should be displayed following User Logout. Valid values are “Yes” or “No”.

2.8.8.3 Request Body JSON Format

```
{
  "username": “[username]”,
  "rx_list": [“[receiver_1]”, “[receiver-2]”, ….], // Mandatory, API does nothing if empty array is
given
  "forced": “Yes” | “No”,
  "splash_screen": “Yes” | “No”
}
```

2.8.8.4 Response - Success

```
HTTP/1.1 200 OK
```

```
Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "User [username] has successfully logged out to the target receivers."
}
```

2.8.8.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- One or more receivers included in the rx-list parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) do(es) not exist: *[List of non-exist receiver names]*."
- One or more receivers included in the rx-list parameter are offline. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) to log out from is(are) offline: *[List of offline receiver names]*"
- The KVM user specified by the username parameter does not exist. This happens when the provided user is neither managed by Boxilla, nor active directory option is enabled. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist. "
- User logout operation fails on one or more receivers due to internal issues. The associated error message in the response body is as follows:
  "**message**": "User logout failed on the following receiver(s): *[List of receiver names where logout failed]*. Other users are successfully logged out."

2.8.8.6 Response - Error: Logging out not Supported by Receiver Firmware

This error response indicates that the firmwares on one or more receivers do not support the associated user-logoff functionality required by this API.

```
HTTP/1.1 403 Forbidden
```

```
Accept: application/json;
Content-Type: application/json;
Accept-version: v1

Body:
{
  "code": "403",
  "message": "Logout via northbound REST API is not supported by the following receiver(s): [List of receivers not supporting user logout]."
}
```

2.8.8.7 Response - Error: Logging out with Force-Logout Disabled while Connections associated to User are Active

This error response indicates that there are active connections associated to the user exist on one or more receivers the user attempts to log out from, when force-logout is disabled.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1

Body:
{
  "code": "403",
  "message": "Active connections associated to user [username] exist in one or more target receivers."
}
```

2.8.8.8 Response - Error: User not Logged in Receivers

This error response indicates that the KVM user specified by the username parameter is not logged in to one or more target receivers.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1

Body:
{
  "code": "403",
  "message": "User [username] is not logged in to one or more of the following receivers: [List of receiver names the user is not logged in]."
}
```

2.8.9 API for Editing User Favorites as Global or to Zones

This API is to edit the favorite settings for an existing KVM user in Boxilla, whose scope can be global or within a specific zone.

2.8.9.1 Request - Uri

Uri	bxa-api/users/kvm/favs
Method	POST

2.8.9.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Name of the KVM user with the favorite settings to be edited.
	scope	String	Scope of the favorite settings for the KVM user. Valid values include: ”” (empty string value) or a valid name of an existing zone in Boxilla. If scope is set as “”, the user favorite is configured as global in Boxilla and not assigned to any individual zone.
	settings	JSON object	Details of favorite settings for this user, containing a list of key/value pairs: • key: the index number of the favorite hotkey where the connection is allocated in the favorite, string format with valid numeric values from “0” – “9” • value: name of connection allocated to the related hotkey number, string format, can be empty value indicating that the connection is unallocated

2.8.9.3 Request Body JSON Format

```
{
  "username": “[username]”,
  "scope": “” | “[zone_1]” | “[zone_2]” ….,
  "settings": {
    "[favorite/hotkey_number]": [name of the connection allocated to hotkey],
    … // Potentially multiple settings key/value pairs, up to 10 with key ranging from “0” – “9”
  }
}
```

The following example of the request body shows how the favorite numbers and connection names are matched with each other in the "settings" parameter:

```
{
  "username": "user1",
  "scope": "zone1",
  "settings": {
    "0": "connection1",
    "3": "connection4"
  }
}
```

The example request above allocates connection "connection1" to hotkey 0 and connection "connection4" to hotkey 3, and assigns the user favorite to zone "zone1" for user "user1".

2.8.9.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
 "code": "200",
 "message": "Successfully updated favorite settings for user [username] to [scope]."
}
```

2.8.9.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- No KVM user exists with the specified username parameter in the request body. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist."

- No zone exists with the specified scope parameter in the request body, if the value is not global. The associated error message in the response body is as follows:
  "**message**": "Zone *[scope]* does not exist for favorite settings for user *[username]*."
- One or more connections in the favorite settings JSON object do not exist. The associated error message in the response body is as follows:
  "**message**": "One or more connections do not exist."
- One or more connections in the favorite settings do not belong to the specified user. The associated error message in the response body is as follows:
  "**message**": "One or more connections in favorite settings do not belong to user [username]."

2.8.9.6 Response - Error: Connection not Assigned to Zone

This error response indicates that one or more connections specified in the settings parameter are not assigned to the specified zone.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Connection [connection_name] is not in zone [zone_name]."
}
```

2.8.9.7 Response - Error: Favorite Settings Exceeding Limit

This error response indicates that the number of favorite settings exceeds limit of 10, as favorite number is limited to 0-9.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Favorite settings exceeding limit."
}
```

2.8.9.8 Response - Error: Duplicate Favorite Number for Multiple Connections

This error response indicates that within the settings JSON object array parameter, multiple different connections are allocated to duplicate favorite number.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Duplicate favorite number detected in request with multiple connection allocated."
}
```

2.8.9.9 Response - Error: Duplicate Connections for Multiple Favorite Numbers

This error response indicates that within the settings JSON object array parameter, duplicate connections are allocated to multiple different favorite numbers.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Duplicate connection detected in request allocated to multiple favorite numbers."
}
```

2.9 APIs for Device-oriented Operations

The APIs for device operations include functionalities as follows:

- Get KVM receivers/transmitters
- Get all properties set on each KVM appliance
- Get KVM appliance status and information

- Set properties of KVM appliances
- Reboot specific/all KVM appliances

The associated design of each API is listed as follows:

2.9.1 API for Getting KVM Appliances

This API is to obtain the list of all the receivers/transmitters managed on the host Boxilla. Valid property parameters are listed in the parameter table below.

2.9.1.1 Request - Uri

Uri	bxa-api/devices/kvm
Method	GET

2.9.1.2 Request – Params

Requirement	Name	Type	Description
Optional	device_type	String	Type of device. Valid values are “transmitter” or “receiver”.
	device_names	String array	List of names of KVM appliances to be obtained.

Note:

- **If device_names parameter is null, the operation would be considered as getting all KVM appliances and the response is returned accordingly.**
- **If device_names parameter is an empty array, the API request is successful but the “devices” JSON array in the response body would be empty.**
- **If device_names is not null and contains valid device names, the value of device_type parameter would be ignored.**
- **If device_names is null and device_type is “transmitter” or “receiver”, the response would only contain the list of all the transmitters/receivers, respectively.**

2.9.1.3 Request Body JSON Format

```
{
  "device_type": “receiver” | “transmitter”,              // Optional, can be null
  "device_names": [“[device_1]”, “[device_2]”, ….]       // Optional, can be null or empty array
}
```

2.9.1.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
    "devices": [
      {
        "name": "[device_name]",
        "ip": "[ip]",
        "mac": "[MAC address]",
        "model": "[sales part number]",
        "swversion": "[firmware version]",
        "serialno": "[serial number]"
      },
      ....   // Multiple appliances
    ]
  }
}
```

2.9.1.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"
- "device_names" parameter value is not an array or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter device_names: *[device_names]*, expecting a String array."
- One or more KVM appliances specified in the device_names parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following device(s) do(es) not exist: *[List of device_names do not exist].*"

2.9.2 API for Getting KVM Appliance Properties

This API is to obtain the property settings of each KVM appliance managed by the host Boxilla. Valid property parameters are listed in the parameter table below.

2.9.2.1 Request - Uri

Uri	bxa-api/devices/kvm/properties
Method	GET

2.9.2.2 Request – Params

Requirement	Name	Type	Description
Optional	device_names	String array	List of names of the KVM devices managed in Boxilla.

Note:

- **If device_names parameter is null, the operation would be considered as getting all KVM appliance properties and the response is returned accordingly.**
- **If device_names parameter is an empty array, the API request is successful but the "properties" JSON array in the response body would be empty.**

2.9.2.3 Request Body JSON Format

```
{
  "device_names": ["[device-1]", "[device-2]", ….]        // Optional, can be null or empty array
}
```

2.9.2.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": {
    "properties": [
      {
        "device_name": "[device_name]",
        "property_state": "Configured" | "Waiting" | "Configuring",
        "model": "[sales part number]",
        "swversion": "[firmware version]",
      "zone": "[name of the zone to which the receiver is assigned]"  // If device is receiver,
available from API version 1.1
      [List of supported properties]
     },
     {….}, // If multiple properties
     ….
   ]
  }
}
```

Note:

In the response body, the "list of supported properties" tag composes of the properties supported by the requested KVM device, which may be different based on the manufactory brand/class and the firmware version of the KVM device. **The full supported list of properties for various types of KVM devices is listed in table 2 in the appendix section 4.1**.

The following three examples show cases of the JSON element in "properties" JSON array in the response body, with various values of keys determining the type of device.

Example 1:

```
{
  "device_name": "transmitter-A",
  "property_state": "Configured",
  "model": "EMD2002SE-T",
  "swversion": "V5.3.1_r5666",
  "video_quality": "Default",
  "video_source_optimization": "Off",
  "hid_configurations": "3",
  "mouse_keyboard_timeout": "0",
  "edid_settings_dvi1": "1920x1080",
  "edid_settings_dvi2": "1920x1200"
}
```

Example 2:

```
{
  "device_name": "transmitter-B",
  "property_state": "Configured",
  "model": "EMD4000T",
  "swversion": "V1.3.1_r5639",
  "hid_configurations": "3",
  "mouse_keyboard_timeout": "0"
}
```

Example 3:

```
{
  "device_name": "receiver-A",
  "property_state": "Configured",
  "model": "EMD2000SE-R",
  "swversion": "V5.4.0_r6438",
  "zone": "zone 1",  // available from API version 1.1
  "config": "Unique",
  "http_enabled": "Enabled"
}
```

2.9.2.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"
- "device_names" parameter value is not an array or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter device_names: *[device_names]*, expecting a String array."
- One or more KVM appliances specified in the device_names parameter has corrupted type-related properties. The associated error message in the response body is as follows:
  "**message**": "Invalid property configuration for device *[name of the device]*: property conflict or missing."
- One or more KVM appliances specified in the device_names parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following device(s) do(es) not exist: *[List of device_names do not exist].*"

2.9.3 API for Setting KVM Receiver Properties

This API is to update the unique property settings of individual KVM receiver managed by the host Boxilla. Valid property parameters are listed in the parameter table below.

2.9.3.1 Request - Uri

Uri	bxa-api/devices/kvm/properties/rx
Method	PUT

2.9.3.2 Request – Params

Requirement	Name	Type	Description
Mandatory	device_name	String	Name of the KVM receiver managed in Boxilla. Maximum length 255.
	http_enabled	String	HTTP-enabled property to set. Valid values are “Enabled” or “Disabled”.
	zone	String	Name of the zone this receiver is assigned to. Maximum length 32. **This parameter is only applicable from API version 1.1 or newer. If parameter value is empty string, receiver would be unassigned from the zone it is currently assiigned to (if any).**

2.9.3.3 Request Body JSON Format

```
{
  "device_name": “[device_name]”,
  "http_enabled": “Enabled” | “Disabled”,
  "zone": “[name of the zone to which the receiver is assigned]”  // can be empty string value
}
```

2.9.3.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": “Receiver settings have been updated successfully.”
}
```

2.9.3.5 Response – Error: bad request

This error response may be caused by:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"
- The specified device has corrupted type-related properties. The associated error message in the response body is as follows:
  "**message**": "Unknown type of device *[device_name]*. Device properties might be missing or corrupted."
- Receiver is not found on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Device *[device_name]* does not exist."
- The property parameter values in the API request body are invalid or incompatible. The associated error message in the response body is as follows:
  "**message**": "Parameters *[list of params]* are invalid or incompatible with firmware of receiver *[device_name]*."
- Receiver is offline. The associated error message in the response body is as follows:
  "**message**": "Device *[device_name]* is offline. "
- Operation fails on the receiver due to internal issues of the device firmware. The associated error message in the response body is as follows:
  "**message**": "Device *[device_name]* is currently not functional."
- The zone to be assigned does not exist on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone]* does not exist."

2.9.3.6 Response – Error: attempting to set unique properties for receiver configured with non-unique properties

This error response indicates that the specified receiver in request parameters is configured with template or system properties at the moment. As northbound REST API version 1 supports unique device property configuration only, in this case the operation would be forbidden.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
```

```
Body:
{
  "code": "403",
  "message": "Receiver [device_name] is with template/system properties. Editing is enabled for unique properties only."
}
```

2.9.3.7 Response – Error: receiver firmware not supporting editing properties

This error response indicates that the version of firmware deployed on the specified receiver in request parameters does not support editing properties. Usually this is caused by older versions of firmware without the associated appliance API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Firmware [firmware version] of [receiver model] receiver [device_name] does not support editing settings."
}
```

2.9.4 API for Setting KVM Transmitter Properties

This API is to set the property settings of individual KVM transmitter managed by the host Boxilla. Valid property parameters are listed in the parameter table below.

2.9.4.1 Request - Uri

Uri	bxa-api/devices/kvm/properties/tx
Method	PUT

2.9.4.2 Request – Params

Requirement	Name	Type	Description
Mandatory	device_name	String	Name of the KVM transmitter managed in Boxilla. Maximum length 255.

Conditional	video_quality	String	Video quality property to set. Valid values are “Best Quality”, “2”, “Default”, “4” or “Best Compression”. (Case sensitive) Supported by single/dual-head Emerald-SE/Emerald-PE transmitters and ZeroU transmitters.
	video_source_optimization	String	Video source optimization property to set. Valid values are “Off”, “DVI/DP Optimised”, “VGA - High Performance”, “VGA - Optimised” or “VGA - Low Bandwidth”. (Case sensitive) Supported by single-head Emerald-SE/Emerald-PE and ZeroU transmitters.
	hid_configurations	String	HID property to set. Valid values are integer formatted strings from 0-5, which represents: 0 – Default; 1 – Basic; 2 – MAC; 3 – Absolute; 4 – Absolute MAC; 5 – Dual Mouse 0-3 are supported by all types of transmitters. 4 is supported by single/dual-head Emerald-SE/Emerald-PE and ZeroU with firmware version no older than 5.3.0, and Emerald-4K with firmware version no older than 1.3.1. 5 is supported by single/dual-head Emerald-SE/Emerald-PE and ZeroU with firmware version no older than 5.4.1, and Emerald-4K with firmware version no older than 1.3.1.
	mouse_keyboard_timeout	String	Mouse-keyboard timeout property to set. Valid values are integer formatted strings from 0-5, representing timeout in seconds. Supported by all single/dual-head Emerald-SE/Emerald-PE and ZeroU transmitters, and Emerald-4K transmitters with firmware version no older than 1.4.0.
	edid_settings_dvi1	String	EDID DVI property to set for head 1 on transmitter. Valid values are “1920x1080”, “1920x1200”, “1680x1050”, “1280x1024” and “1024x768” for Emerald-SE/PE/ZeroU transmitters, and “Default”, “3840x2160p-60Hz”, “3840x2160p-30Hz”, “2560x1440p-60Hz”, “1920x1080p-60Hz” for Emerald-4K transmitters. (Case sensitive) Supported by all single/dual-head Emerald-SE/Emerald-PE and ZeroU transmitters, and Emerald-4K transmitters with firmware version no older than 1.4.0.

	edid_settings_dvi2	String	EDID DVI property to set for head 2 on transmitter (Dual-head only). Valid values are the same with edid_settings_dvi1. Supported by dual-head Emerald-SE and Emerald-PE transmitters.

Note: for EDID DVI properties, this API does not support "Clone" as a valid value, which is different from the API to get transmitter properties.

2.9.4.3 *Request Body JSON Format*

```
// Only "device_name" is mandatory, the rest parameters are conditionally applicable as described above

{
  "device_name": "[device_name]",
  "video_quality": "Best Quality" | "2" | "Default" | "4" | "Best Compression",
  "video_source_optimization": "Off" | "DVI/DP Optimised" | "VGA - High Performance" | "VGA - Optimised" | "VGA - Low Bandwidth",
  "hid_configurations": "0" | "1" | "2" | "3" | "4" | "5",
  "mouse_keyboard_timeout": "0" | "1" | "2" | "3" | "4" | "5",
  "edid_settings_dvi1": "1920x1080" | "1920x1200" | "1680x1050" | "1280x1024" | "1024x768" (5 possible values for Emerald-SE/PE/ZeroU) | "Default" | "3840x2160p-60Hz" | "3840x2160p-30Hz" | "2560x1440p-60Hz" | "1920x1080p-60Hz" (5 possible values for Emerald-4K),
  "edid_settings_dvi2": "1920x1080" | "1920x1200" | "1680x1050" | "1280x1024" | "1024x768"
}
```

2.9.4.4 *Response - Success*

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "Transmitter settings have been updated successfully."
}
```

2.9.4.5 *Response – Error: bad request*

This error response may be caused by:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"
- The specified device has corrupted type-related propeties. The associated error message in the response body is as follows:
  "**message**": " Unknown type of device *[device_name]*. Device properties might be missing or corrupted."
- Transmitter is not found on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Device *[device_name]* does not exist. "
- The property parameter values in the API request body are invalid or incompatible. The associated error message in the response body is as follows:
  "**message**": "Parameters *[list of params]* are invalid or incompatible with firmware of transmitter *[device_name]*."

The following response shows an example of incompatible parameters:

```
HTTP/1.1 400 Bad Request
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": “400”,
  "message": "Parameters ‘video_quality’ and ‘video_source’ are invalid or incompatible with firmware of transmitter ‘Emerald4K-TX-1’."
}
```

2.9.4.6 Response – Error: attempting to set unique properties for transmitter configured with non-unique properties

This error response indicates that the specified transmitter in request parameters is configured with template or system properties at the moment. As northbound REST API version 1 supports unique device property configuration only, in this case the operation would be forbidden.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
```

```
Accept-version: v1

Body:

{
  "code": "403",
  "message": "Transmitter [device_name] is with template/system properties. Editing is enabled
for unique properties only."
}
```

2.9.4.7 Response – Error: transmitter firmware not supporting editing properties

This error response indicates that the version of firmware deployed on the specified transmitter in request parameters does not support editing properties. Usually this is caused by older versions of firmware without the associated appliance southbound REST API.

```
HTTP/1.1 403 Forbidden

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "403",
  "message": "Firmware [firmware version] of [transmitter model] transmitter [device_name] does
not support editing settings."
}
```

2.9.5 API for Setting KVM Receiver Zone assignment

This API is to update the zone the specified receiver is assigned to.

2.9.5.1 Request - Uri

Uri	bxa-api/devices/kvm/rx/zones
Method	POST

2.9.5.2 Request – Params

Requirement	Name	Type	Description
Mandatory	device_name	String	Name of the KVM receiver managed in Boxilla. Maximum length 255.
	zone	String	Name of the zone this receiver is assigned to. Maximum length 32.

			If provided with an empty string value, receiver would be unassigned from the zone currently assigned to (if any).

2.9.5.3 Request Body JSON Format

```
{
  "device_name": "[device_name]",
  "zone": "[name of the zone to which the receiver is assigned]"
}
```

2.9.5.4 Response – Success Reassignment to New Specified Zone

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  "code": "200",
  "message": "Receiver [device_name] has been successfully reassigned to zone [zone]."
}
```

2.9.5.5 Response – Success Unassignment to Original Zone (if any)

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  "code": "200",
  "message": "Receiver [device_name] has been successfully unassigned from zone
[original_zone]."
}
```

2.9.5.6 Response – Error: bad request

This error response may be caused by:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:

"**message**": "Invalid parameter(s): *[List of parameters detected as invalid]*."

- Receiver is not found on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Receiver *[device_name]* does not exist."
- The zone to be assigned does not exist on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone]* does not exist."
- Receiver is offline. The associated error message in the response body is as follows:
  "**message**": "Receiver *[device_name]* is offline."
- Reassignment of receiver to zone fail due to the embedded software issue. The associated error message in the response body is as follows:
  "**message**": "Receiver *[device_name]* is currently not functinoal."

2.9.6 API for Getting KVM Appliance Status

This API is to obtain the real-time status of one or more KVM appliances managed by the host Boxilla, including the following contents:

- Information of logged-in users (receiver-only)
- Information of the zone it is assigned to. **This is only applicable if the device is a receiver and assigned to a zone, for API version >= 1.1**.
- Information of active connections
- Physical online/offline status
- Appliance setting update status:
  - Waiting: setting update is initializing.
  - Configuring: setting update is processing.
  - Configured: setting update is complete.
  - **Note: as editing appliance settings using APIs in section 2.8.3 and 2.8.4 involve communications with appliances and may take extra time, this GET API should be used to monitor the configuration state of appliance settings as needed.**
- Appliance details:
  - Basic: name; model number; serial number; firmware version
  - A unique identifier generated internally by individual Boxilla instance at the initialization state of managing the device
  - Operational details: total/free memory/storage; uptime; load average in 1/5/15 minutes

- Network details: mac; ip; netmask; gateway; broadcast; primary/secondary DNS; multicast-ip and master port

Valid property parameters are listed in the parameter table below.

2.9.6.1 Request - Uri

Uri	bxa-api/devices/kvm/status
Method	GET

2.9.6.2 Request – Params

Requirement	Name	Type	Description
Optional	device_names	String array	List of names of the KVM devices managed in Boxilla.

2.9.6.3 Request Body JSON Format

```
{
  "device_names": ["[device-1]", "[device-2]", ....]      // Optional, can be null or empty array
}
```

2.9.6.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
    "devices": [
     {
      "uid": "[Unique identifier generated internally by Boxilla for each device]",
      "device_type": "receiver" | "transmitter",
      "zone": "[Name of the zone it is assigned to]", // This is applicable only if device is receiver
      "logged_in user": {      // Available for request to receivers only, empty if no user logged in
        "username": "[logged-in username]",
        "type": "active directory user" | "common user"
      },
      "active_connections": [
        {
```

```
       "active_connection_name": "[active_connection_name]",
       "active_connection_type": "Private" | "Shared",
       "active_connection_group": "ConnectViaTx" | "VM" | "VMPool" | "VMHorizon" | "TXPair",
      },
      {….},   // if there are more active connections
      ….
    ],
    "state": "Online" | "Offline",
    "status": "Waiting" | "Configuring" | "Configured",
    "device_name": "[appliance-name]",
    "mac": "[MAC address]",
    "ip": "[ip]"
   },
   {….},  // if there are more appliances
   ….
  ]
 }
}
```

2.9.6.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.) The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- One or more KVM devices specified by the device_names parameter do not exist on Boxilla. The associated error message in the response body is as follows:
  "**message**": "The following device(s)do(es) not exist: *[non_exist_device_names]*. "
- One or more KVM devices specified by the parameter are currently offline. The associated error message in the response body is as follows:
  "**message**": "*[offline_device_names]* currently offline. "

2.9.7 API for Rebooting all KVM Appliances

This API is to send reboot command to all KVM appliances managed by the host Boxilla.

2.9.7.1 Request - Uri

Uri	bxa-api/devices/kvm/reboot-all

Method	POST

2.9.7.2 Request – Params

This API does not require any parameters within the request body. The version of API is required to be included in the request header.

2.9.7.3 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": “All KVM devices are successfully rebooted.”
}
```

2.9.7.4 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- One or more KVM devices to be rebooted are currently offline. The associated error message in the response body is as follows:
  "**message**": "The following device(s) to reboot is(are) offline: *[offline_device_names]*. "
- Rebooting operation fails on one or more devices due to internal issues. The associated error message in the response body is as follows:
  "**message**": "Fail to reboot the following device(s): *[List of device names where rebooting failed]*."

2.9.7.5 Response - Error: Rebooting via API not Supported by Device Firmware

This error response indicates that the firmwares on one or more devices specified in the parameters do not support the associated device rebooting functionality required by this API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
```

```
Body:
{
  "code": "403",
  "message": "Rebooting via northbound REST API is not supported by the following device(s): [List of devices not supporting rebooting]."
}
```

2.9.8 API for Rebooting selected KVM Appliances

This API is to send reboot command to individual KVM appliance managed by the host Boxilla. Valid parameters are listed in the parameter table below.

2.9.8.1 *Request - Uri*

Uri	bxa-api/devices/kvm/reboot
Method	POST

2.9.8.2 *Request – Params*

Requirement	Name	Type	Description
Optional	device_names	String array	List of names of KVM devices managed in Boxilla to be rebooted.

2.9.8.3 *Request Body JSON Format*

```
{
  "device_names": ["[device-1]", "[device-2]", ….],       // Optional, can be null or empty array
}
```

2.9.8.4 *Response - Success*

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "The selected KVM device(s) is(are) successfully rebooted."
}
```

2.9.8.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- “device_names” parameter value is not an array or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter device_names: *[device_names]*, expecting a String array."
- One or more KVM appliances specified in the device_names parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following device(s) do(es) not exist: *[List of device_names do not exist].*"
- One or more KVM devices to be rebooted are offline. The associated error message in the response body is as follows:
  "**message**": "The following device(s) to reboot is(are) offline: *[offline_device_names]*. "
- Rebooting operation fails on one or more devices due to internal issues. The associated error message in the response body is as follows:
  "**message**": "Fail to reboot the following device(s): *[List of device names where rebooting failed]*."

2.9.8.6 Response - Error: Rebooting via API not Supported by Device Firmware

This error response indicates that the firmwares on one or more devices specified in the parameters do not support the associated device rebooting functionality required by this API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1

Body:

{
  "code": "403",
  "message": "Rebooting via northbound REST API is not supported by the following device(s): [List of devices not supporting rebooting]."
}
```

2.10 APIs for Connection-oriented Operations

The APIs for connection-related operations include the following functionalities:

- Create new KVM connections between multiple BlackBox entities (transmitters/receivers, VMs, etc. that exchange traffic)
- Edit existing KVM connection properties
- Deleting existing KVM connection records
- Make/break KVM active connections

In Boxilla, each KVM connection active record is categorized into the following list of groups, based on the type of traffic source in the connection:

- ConnectViaTx: the traffic source is a transmitter device
- VM: the traffic source is a VM
- VMPool: the traffic source is a group of VMs
- VMHorizon: the traffic source is a VM with horizontal view
- TXPair: dual traffic sources exist from multiple transmitters

Creating/editing of each group of KVM connection is designed with the specific set of properties for as input parameters, whose associated requirements are defined as follows:

- Mandatory: the parameter is required
- Unique Params: the parameter is required when creating/editing unique connections, including the following categories:
  - ConnectViaTx Params: the parameter is optionally applicable for viaTx connections
  - VM Params: the parameter is optionally applicable for VM connections
  - VMPool Params: the parameter is optionally appliable for VM pool connections
  - VMHorizon Params: the parameter is optionally applicable for VM horizontal connections
  - TXPair Params: the parameter is optionally applicable for TXPair connections

The following table defines the conditional parameters for each category of unique connection:

Requirement	Name	Type	Description
viaTx Params	extended_desktop	String	Enabled/disabled option for extended desktop configuration. Valid values are “Yes” or “No”. (Case sensitive)
	usb_redirection	String	Enabled/disabled option for USB redirection. Valid values are “Yes” or “No”. (Case sensitive)

	audio	String	Enabled/disabled option for audio. Valid values are “Yes” or “No”. (Case sensitive)
	persistent	String	Enabled/disabled option for connection persistency. Valid values are “Yes” or “No”. (Case sensitive)
	cmode	String	Compression mode supported for 2K-4K compatibility feature. Valid values are “0” or “10”. “0” represents lossless mode and “10” represents optimized mode. **Lossless mode 0 is valid only when the host of the connection is an Emerald-4K transmitter**.
VM Params	username	String	Username credential used for BlackBox VM login. **Optional**. Valid value needs to be the username of an existing user.
	password	String	Password credential used for BlackBox VM login. **Optional unless “username” is provided**. Valid value needs to be in pair with a valid username.
	extended_desktop	String	Enabled/disabled option for extended desktop configuration. Valid values are “Yes” or “No”. (Case sensitive)
	usb_redirection	String	Enabled/disabled option for USB redirection. Valid values are “Yes” or “No”. (Case sensitive)
	audio	String	Enabled/disabled option for audio. Valid values are “Yes” or “No”. (Case sensitive)
	port	String	Port number (in numeric format) of the VM connected to (if applicable). Maximum length 5.
	domain	String	Domain name of the VM connected to (if applicable). Maximum length 255.
	nla	String	Enabled/disabled option for NLA. Valid values are “Yes” or “No”. (Case sensitive)
VM-Pool Params	extended_desktop	String	Enabled/disabled option for extended desktop configuration. Valid values are “Yes” or “No”. (Case sensitive)
	usb_redirection	String	Enabled/disabled option for USB redirection. Valid values are “Yes” or “No”. (Case sensitive)
	audio	String	Enabled/disabled option for audio. Valid values are “Yes” or “No”. (Case sensitive)
VM-Horizon Params	username	String	Username credential used for BlackBox VM login. **Optional**. Valid value needs to be the username of an existing user.
	password	String	Password credential used for BlackBox VM login. **Optional unless “username” is provided**. Valid value needs to be in pair with a valid username.
	protocol	String	Protocol used for VM Horizon view connections. Valid value is “PCoIP” only. (Case sensitive)

TxPair Params	host_2	String	IP/DNS address of the secondary traffic source of the KVM connection.
	audio	String	Enabled/disabled option for audio. Valid values are “Yes” or “No”. (Case sensitive)
	persistent	String	Enabled/disabled option for connection persistency. Valid values are “Yes” or “No”. (Case sensitive)
	pairing_type	String	Pairing type specified for TxPair connection. Valid values are “1” or “2”.
	orientation	String	Orientation of dual paired views of TxPair connections. Required only when “**pairing_type**” is set to “2”. Valid values are ”H12”, “H21”, “V12” or “V21” (Case sensitive), where “H/V” stands for Horizontal/Vertical and “12/21” represents destination from view 1 to view 2 or reverse.
	audio_source	String	Address of the audio source. Required only when “**audio**” is set as “Yes”. Valid values are “1” or “2”, representing whether the audio source is specified as the value of host or host_2.

Table 1. KVM Connection Property Dependency based on Group of Connection (viaTx/VM/VMPool/VMHorizon/TxPair)

The details of the API for each group of KVM connection are introduced in the sections below, respectively.

2.10.1 API for Creating KVM Connections

This API is to create a new KVM connection with type, group and properties specified within the parameters in request body.

2.10.1.1 Request - Uri

Uri	bxa-api/connections/kvm
Method	POST

2.10.1.2 Request - Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the KVM connection. Maximum length 64.
	host	String	IP/DNS address of the primary traffic source of the connection. Value cannot be set if “group” is “VMPool”.
	group	String	Connection group category. Valid values are: “ConnectViaTx” | “VM” | “VMPool” | “VMHorizon” | “TXPair”. (Case sensitive)

	connection_type	String	Connection type. Valid values are "Private", "Shared" or "Exclusive". (Case sensitive) **Value is fixed to be "Private" when "group" is set as "VM", "VMPool" or "VMHorizon".** **Type "Exclusive" is only valid for ConnectViaTx connections, any other groups of connections will not supported.**
	view_only	String	Enabled/disabled view-only option. Valid values are "Yes" or "No". (Case sensitive)
	zone	String	Name of the zone this connection is assigned to. Maximum length 32. **If provided with empty string value, connection is not assigned to any zone.**
viaTx Params			Required when "group" is "ConnectViaTx". **"usb_redirection" parameter value cannot be "Yes" if "connection_type" is "Shared".** **"cmode" parameter value can be "0" only when the host of the connection is an Emerald-4K transmitter.** **"view_only" parameter value can only be "No" if "connection_type" is "Exclusive".**
VM Params			Required when "group" is "VM".
VM-Pool Params			Required when "group" is "VMPool".
VM-Horizon Params			Required when "group" is "VMHorizon".
TxPair Params			Required when "group" is "TXPair".

2.10.1.3 Request Body JSON Format – viaTx Connections

```
{
  "name": "[connection_name]",
  "host": "[host]",
  "group": "ConnectViaTx",
  "connection_type": "Private" | "Shared",
  "extended_desktop": "Yes" | "No",
  "usb_redirection": "Yes" | "No",
  "audio": "Yes" | "No",
  "persistent": "Yes" | "No",
  "view_only": "Yes" | "No",
  "cmode": "0" | "10",
  "zone": "Zone 1"   // Can be empty string
}
```

2.10.1.4 Request Body JSON Format – VM Connections

```
{
  "name": "[connection_name]",
  "host": "[host]",
  "group": "VM",
  "connection_type": "Private",
  "username": "[VM-login username]",
  "password": "[VM-login password]",
  "extended_desktop": "Yes" | "No",
  "usb_redirection": "Yes" | "No",
  "audio": "Yes" | "No",
  "port": "[port]",
  "domain": "[domain]",
  "nla": "Yes" | "No",
  "view_only": "Yes" | "No"
}
```

2.10.1.5 Request Body JSON Format – VM-Pool Connections

```
{
  "name": "[connection_name]",
  "host": "[host]",
  "group": "VMPool",
  "connection_type": "Private",
  "extended_desktop": "Yes" | "No",
  "usb_redirection": "Yes" | "No",
  "audio": "Yes" | "No",
  "view_only": "Yes" | "No"
}
```

2.10.1.6 Request Body JSON Format – VM-Horizon Connections

```
{
  "name": "[connection_name]",
  "host": "[host]",
  "group": "VMHorizon",
  "connection_type": "Private",
  "username": "[VM-login username]",
  "password": "[VM-login password]",
  "protocol": "PCoIP",
  "view_only": "Yes" | "No"
}
```

2.10.1.7 Request Body JSON Format – TxPair Connections

```
{
  "name": "[connection_name]",
  "host": "[host]",
  "host_2": "[host_2]",    // Optional
  "group": "TXPair",
  "connection_type": "Private" | "Shared",
  "audio": "Yes" | "No",
  "persistent": "Yes" | "No",
  "view_only": "Yes" | "No",
  "pairing_type": "1" | "2",
  "orientation": "H12" | "H21" | "V12" | "V21",   // applicable only when "pairing_type" is "2"
  "audio_source": "1" | "2"        // applicable only when "audio" is "Yes"
}
```

2.10.1.8 Response - Success

```
HTTP/1.1 201 Created

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "201",
  "message": "Created [group] connection [name]."
}
```

2.10.1.9 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Property-related parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- The zone to be assigned does not exist on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone]* does not exist."
- When creating TX pair connection while one or more hosts are not from single head transmitters, the message body is as follows:

"**message**": "*TX pair connection only support single head appliances (Host [1/2])*."

- Multiple property-related parameters are not compatible with each other, e.g. passing the *connection_type* parameter as "shared" together with the group parameter as "VM, "VMPool" or "VMHorizon". (Check the parameter table for this API for details). The associated error message in the response body is as follows:
  "**message**": "Parameters *[List of incompatible parameters]* are incompatible with *[group]* connection *[name]*."
  The following response shows an example of incompatible parameters:

```
HTTP/1.1 400 Bad Request
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "400",
  "message": "Parameters 'connection_type': 'Shared' are incompatible with VM connection
'connection-1'."
}
```

2.10.1.10 Response - Error: License-related Failure

This error response indicates that the Boxilla license installed on the server is invalid, expired or limitation of the maximum number of users is reached.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "License limit reached."
}
```

2.10.1.11 Response - Error: Connection Already Exists

This error response indicates that a KVM connection with the name specified in request body already exists and the API is attempting to create a duplicate record.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Duplicate name received."
}
```

2.10.2 API for Editing KVM Connection Profiles/Properties

This API is to edit the profile/properties of an existing KVM connection whose type, group and properties are specified within the parameters in request body.

2.10.2.1 Request - Uri

Uri	bxa-api/connections/kvm
Method	PUT

2.10.2.2 Request - Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the KVM connection. Maximum length 64.
	host	String	IP/DNS address of the primary traffic source of the connection. Value cannot be set if "group" is "VMPool".
	group	String	Connection group category. Valid values are: "ConnectViaTx" | "VM" | "VMPool" | "VMHorizon" | "TXPair". (Case sensitive)
	connection_type	String	Connection type. Valid values are "Private", "Shared" or "Exclusive". (Case sensitive) **Value is fixed to be "Private" when "group" is set as "VM", "VMPool" or "VMHorizon".** **Type "Exclusive" is only valid for ConnectViaTx connections, any other groups of connections will not supported.**
	view_only	String	Enabled/disabled view-only option. Valid values are "Yes" or "No". (Case sensitive)
	zone	String	Name of the zone this connection is assigned to. Maximum length 32.

			This parameter is only applicable from API version 1.1 or newer. **If parameter is provided with empty string, connection would be unassigned from the zone it is currently assigned to (if any).**
Optional	new_name	String	New name of the KVM connection to be edited. Maximum length 64. Connection name is updated if provided.
viaTx Params			Required when “group” is “ConnectViaTx”. **“usb_redirection” parameter value cannot be “Yes” if “connection_type” is “Shared”.** **“cmode” parameter value can be “0” only when the host of the connection is an Emerald-4K transmitter.** **“view_only” parameter value can only be “No” if “connection_type” is “Exclusive”.**
VM Params			Required when “group” is “VM”.
VM-Pool Params			Required when “group” is “VMPool”.
VM-Horizon Params			Required when “group” is “VMHorizon”.
TxPair Params			Required when “group” is “TXPair”.

2.10.2.3 Request Body JSON Format

For each group of connection (viaTx, VM, VM-Pool, VM-Horizon or TxPair), the request is individually defined in the same JSON format specified in section 2.9.1.3 - 2.9.1.7, respectively, with the additional “new_name” parameter available. If the “new_name” parameter is specified with a valid value (name-format string; max-length 64; obeying the naming rules defined in section 4.3.1), the name of the connection would be updated with the value.

2.10.2.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": "Updated properties for connection [name]."
}
```

2.10.2.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Property-related parameters are passed in as empty or invalid (contain invalid characters; length exceeding limit; etc.). The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- Multiple property-related parameters are not compatible with each other, e.g. passing the *connection_type* parameter as “shared” together with the group parameter as “VM, “VMPool” or “VMHorizon”. (Check the parameter table for this API for details). The associated error message in the response body is as follows:
  "**message**": "Parameters *[List of incompatible parameters]* are incompatible with *[group]* connection *[name]*."
- The associated KVM connection record does not exist. The associated error message in the response body is as follows:
  "**message**": "Connection *[name]* does not exist."
- The zone to be assigned does not exist on Boxilla. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone]* does not exist."
- When updating TX pair connection while one or more hosts are not from single head transmitters, the message body is as follows:
  "**message**": "*TX pair connection only support single head appliances (Host [1/2])*."

2.10.2.6 Response – Error: Attempting to Edit Properties of non-Unique Connections

This error response indicates that the specified connection is with non-unique properties. Editing properties of the connection would be prohibited by the northbound REST API and a HTTP 403 error response is returned. The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": " Editing properties for non-unique connection [name] is not supported."
}
```

2.10.2.7 Response – Error: Changing Connection Name to Name of Existing Connection

This error response indicates that the new connection name to be changed to belongs to an existing connection. The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Cannot update connection name to [new_name]. Connection [new_name] already
exists."
}
```

2.10.3 API for Deleting all KVM Connections

This API is to delete all the existing KVM connections with unique properties in Boxilla.

2.10.3.1 Request - Uri

Uri	bxa-api/connections/kvm/all
Method	DELETE

2.10.3.2 Request – Params

This API does not require any parameters within the request body. The version of API is included in the request header.

2.10.3.3 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Successfully deleted all connections."
}
```

2.10.3.4 *Response - Error: Connection is Active*

This error response indicates that one or more KVM connections to be deleted are active with traffic delivery when the request is being operated.

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "409",
  "message": " Fail to delete connection(s) [connection_names] because one or more connections are active."
}
```

2.10.4 API for Deleting Specific KVM Connections

This API is to delete one or more existing KVM connections with unique properties in Boxilla, specified by the request parameter which contains a list of existing connection names to be deleted.

2.10.4.1 *Request - Uri*

Uri	bxa-api/connections/kvm
Method	DELETE

2.10.4.2 *Request - Params*

Requirement	Name	Type	Description
Mandatory	connection_names	String array	List of names of the KVM connections to be deleted. If array is empty with no connection names inside, the API request is successful but no connection would be deleted.

2.10.4.3 *Request Body JSON Format*

```
{
  "connection_names": ["[connection_1]", "[connection_2]", ...]  // Mandatory, can be empty array
}
```

2.10.4.4 *Response - Success*

```
HTTP/1.1 200 OK
```

```
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Successfully deleted connection(s) [connection_names]."
}
```

2.10.4.5 Response – Success with Empty Connection_names Parameter

```
HTTP/1.1 204 No Content
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
```

2.10.4.6 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required parameters are passed in as empty (null parameter, not parameter as an empty array) or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- “connection_names” parameter is not an array (for example, an integer is given instead) or any elements within are not String type. The associated error message in the response body is as follows:
- "**message**": "Invalid parameter connection_names: *[connection_names]*, expecting a String array."
- One or more KVM connections specified in the connection_names parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following connection(s) *[connection_names]* do(es) not exist."

2.10.4.7 Response – Error: Attempting to Delete Connections with non-Unique Properties

This error response indicates that one or more KVM connections specified in the parameters are with non-unique properties. Deleting these connections would be prohibited by the northbound REST API and a HTTP 403 error response is returned. Meanwhile all the other valid connections specified in the parameters would still be successfully deleted.

The associated error message in the response body is as follows:

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "The following connections are with non-unique properties: [List of connection names]."
}
```

2.10.4.8 Response - Error: Connection is Active

This error response indicates that one or more KVM connections to be deleted are active with traffic delivery when the request is being operated.

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "409",
  "message": " Fail to delete connection(s) [connection_names] because one or more connections are active."
}
```

2.10.5 API for Getting KVM Connection Status

This API is for obtaining status and related information of one or more KVM connections in Boxilla, including the following contents:

- Name of user that logs into the KVM appliance of the connection
- Connection type: Private or Shared
- Connection group: ConnectViaTx, VM, VM-pool, VM-horizon or TXPair
- A unique identifier generated internally by individual Boxilla instance at the initialization state of creating the connection

Valid property parameters are listed in the parameter table below.

2.10.5.1 Request - Uri

Uri	bxa-api/connections/kvm
Method	GET

2.10.5.2 Request – Params

Requirement	Name	Type	Description
Optional	connection_names	String array	List of names of the queried KVM connections

Note:

If no parameter is provided by the API request (parameter is null, not an empty array), the operation would be considered as getting status of all KVM connections and the response is returned accordingly.

If the "connection_names" parameter is an empty array, the API request is successful but no connection status is returned in the response (the "connections" JSON array in response body is empty).

2.10.5.3 Request Body JSON Format

```
{
  "connection_names": ["[connection_1]", "[connection_2]", …]     // Optional, can be null or
empty array
}
```

2.10.5.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": {
    "connections": [
      {
        "uid": "[Unique identifier generated internally by Boxilla for each connection]",
        "name": "[connection_name]",
        "host": "[IP/DNS of primary connection source]",
        "connection_type": "Private" | "Shared",
        "group": "ConnectViaTx" | "VM" | "VMPool" | "VMHorizon" | "TXPair",
        "zone": "[name of the zone to which the connection is assigned]",
        "view_only": "Yes" | "No",
        [List of supported properties]
      },
      {….},  // if there are more connections
      ….
    ]
  }
}
```

Note:

In the response body, the "list of supported properties" tag composes of the properties of the requested KVM connection, which may be different based on the group of connection. The full supported list of properties for each group of connection is listed in table 1 at the beginning of section 2.9.

"Password" property for VM/VMHorizon connection would NOT be returned.

The following three examples show cases of the JSON element in "connections" JSON array in response body for various groups of KVM connections.

Example 1:

```
{
   "name": "tx-connection-1",
   "host": "10.0.2.20",
   "connection_type": "Shared",
   "group": "ConnectViaTx",
   "zone": "zone 1",
   "extended_desktop": "Yes",
   "usb_redirection": "No",
   "audio": "Yes",
   "persistent": "Yes",
   "view_only": "No",
   "cmode": "10"
}
```

Example 2:

```
{
  "name": “vm-connection-2”,
  "host": “10.0.2.21”,
  "connection_type": “Private”,
  "group": “VM”,
  "username": “James”,
  "extended_desktop": "Yes",
  "usb_redirection": "No",
  "audio": “Yes”,
  "port": “3389”,
  "domain": “test.com”,
  "nla": “Yes”,
  "view_only": “No”
}
```

Example 3:

```
{
  "name": “txpair-connection-3”,
  "host": “10.0.2.22”,
  "connection_type": “Private”,
  "group": “TXPair”,
  "host_2": "10.0.2.23",
  "audio": “Yes”,
  "persistent": “Yes”,
  "view_only": “No”,
  "pairing_type": “2”,
  "orientation": “H12”,
  "audio-source": 1
}
```

2.10.5.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required property parameters are passed in as invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as invalid].*"

- “connection_names” parameter is not an array (for example, an integer is given instead) or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter connection_names: *[connection_names]*, expecting a String array."
- One or more KVM connections within the connection_name parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following connection(s) do(es) not exist: *[List of connections do not exist]*"

2.10.6 API for Launching KVM Active Connections

This API is for launching a KVM active connection based on an existing KVM connection record. Requests of this API require authentication using the credentials of the REST API user described in section 2.3.2.

Before the operation of launching the connection, the user specified in the request is logged on to the target receiver forcefully by this API.

2.10.6.1 Request - Uri

Uri	bxa-api/connections/kvm/active
Method	POST

2.10.6.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Name of the KVM user logged in to the receiver.
	password	String	Password of the KVM user logged in to the receiver.
	connection_name	String	Name of KVM connection as the active connection traffic source.
	receiver_name	String	Name of the receiver used to launch the active connection.

2.10.6.3 Request Body JSON Format

```
{
  "username": “[username]”,
  "psaaword": “[password]”,
  "connection_name": “[connection_name]”,
  "receiver_name": “[receiver_name]”
}
```

2.10.6.4 Response - Success

```
HTTP/1.1 201 Created

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "201",
  "message": "Active connection [connection_name] is successfully launched."
}
```

2.10.6.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- The source or destination (the receiver and transmitter/VM) does not exist. The associated error message in the response body is as follows:
  "**message**": "Active connection *[connection_name]* or receiver *[receiver_name]* does not exist."
- The user specified in the username parameter does not exist. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist."
- The source or destination (the receiver and transmitter/VM) is offline. The associated error message in the response body is as follows:
  "**message**": "Active connection *[connection_name]* or receiver *[receiver_name]* is currently unreachable."
- The embedded software on the KVM devices is mal-functional. The associated error message in the response body is as follows:
  "**message**": "Active connection *[connection_name]* or receiver *[receiver_name]* is currently not functional."
- When launching TX pair connection while one or more hosts are not managed, the message body is as follows:
  "**message**": "Not all transmitters are managed."
- When launching TX pair connection into a single head receiver, the message body is as follows:
  "**message**": "TX pair connection does not support single head receiver."

- When launching a private ConnectViaTx connections while there is already an active connection to the same transmitter, the message body is as follows:
  "**message**": "Can not launch a private connection as there is another active connection on transmitter: *[transmitter name]*"
- When launching a shared ConnectViaTx connection while there is already an active private connection to the same transmitter, the message body is as follows:
  "**message**": "Can not launch a shared connection in conjunction with a private connection on transmitter: *[transmitter name]*"
- When launching an Exclusive connection where there is another private or Exclusive connection to the same transmitter, the message body is as follows:
  "**message**": "Can not launch an exclusive connection in conjunction with private connection or exclusive connection on transmitter: *[transmitter name]*."

2.10.6.6 Response - Error: UnAuthorized

This error response may be caused by:

- The REST user credentials included in the request header are invalid, as described in section 2.4.2.
- The user is a local user created on Boxilla and the login password provided in request parameters is incorrect. The associated error message in the response body is as follows:
  "**message**": "Operation is not authorized due to invalid login credentials."
- The user is an active directory user managed by active directory server and the login credentials for it are incorrect. The associated error message in the response body is as follows:
  "**message**": " Operation is not authorized. Active directory user authentication failed on one or more target receivers. Please check the active directory server configuration on your Boxilla and make sure the credentials are correct."

2.10.6.7 Response - Error: Launching Active Connection not Supported by Device Firmware

This error response indicates that the version of firmware deployed on the specified receiver in request parameters does not support launching active connections. Usually this is caused by older versions of firmware without the associated appliance southbound REST API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
```

```
Accept-version: v1
Body:
{
  "code": "403",
  "message": " Launching active connection is not supported by firmware type [receiver model] version [firmware version] of receiver [receiver_name]."
}
```

2.10.6.8 Response - Error: Connection not Assigned to Logged-in User

This error response indicates that the source KVM connection to be launched is not assigned to the logged-in KVM user on the receiver.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Connection [connection_name] not assigned to user [username]."
}
```

2.10.6.9 Response - Error: User Login Timeout on Receiver

This error response indicates that a timeout has occurred during the specified user login on the receiver.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "User [username] login timeout on receiver [receiver_name]."
}
```

2.10.7 API for Terminating KVM Active Connections

This API is to break an existing KVM active connection between a source/destination pair. Requests of this API require authentication using the credentials of the REST API user described in section 2.3.2.

2.10.7.1 Request - Uri

Uri	bxa-api/connections/kvm/active
Method	DELETE

2.10.7.2 Request - Params

Requirement	Name	Type	Description
Mandatory	username	String	Name of the KVM user logged in to the receiver.
	connection_name	String	Name of KVM connection as the active connection traffic source.
	receiver_name	String	Name of the receiver used to terminate the active connection
Optional	splash_screen	String	String indicating if the splash screen should be displayed following Connection Termination. Valid values are “Yes” or “No”.

2.10.7.3 Request Body JSON Format

```
{
  "username": “[username]”,
  "connection_name": “[connection_name]”,
  "receiver_name": “[receiver_name]”,
  "splash_screen": “Yes” | “No”
}
```

2.10.7.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": "Active connection [connection_name] is successfully terminated."
}
```

2.10.7.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- The source or destination (the receiver and transmitter/VM) does not exist. The associated error message in the response body is as follows:
  "**message**": "Active connection *[connection_name]* or receiver *[receiver_name]* does not exist."
- The user specified in the username parameter does not exist. The associated error message in the response body is as follows:
  "**message**": "User *[username]* does not exist."
- Receiver is offline. The associated error message in the response body is as follows:
  "**message**": "Receiver *[receiver_name]* is unreachable."
- The embedded software on the KVM device is mal-functional. The associated error message in the response body is as follows:
  "**message**": "Active connection *[connection_name]* or receiver *[receiver_name]* is currently not functional."

2.10.7.6 Response - Error: Terminating Active Connection not Supported by Device Firmware

This error response indicates that the version of firmware deployed on the specified receiver in request parameters does not support terminating active connections. Usually this is caused by older versions of firmware without the associated appliance API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": " Terminating active connection is not supported by firmware type [receiver model] version [firmware version] of receiver [receiver_name]."
}
```

2.10.7.7 Response - Error: Connection not Active

This error response indicates that the specified connection in the parameter to be terminated is not active at the moment.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Connection [connection_name] is not active."
}
```

2.10.7.8 Response - Error: Connection not Assigned to Logged-in User

This error response indicates that the source KVM connection to be launched is not assigned to the logged-in KVM user on the receiver.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "403",
  "message": "Connection [connection_name] is not assigned to user [username]."
}
```

2.10.8 API for Getting All KVM Active Connection Status

This API is for obtaining status and related information of all existing KVM active connections in Boxilla, including the following contents:

- Connection source/destination names
- Name of logged-in user
- Connection type: private or shared
- Active duration
- Video/audio/USB and total bandwidth
- RTT and user latency
- FPS/dropped-FPS
- Status of audio/USB

2.10.8.1 Request - Uri

Uri	bxa-api/connections/kvm/active
Method	GET

2.10.8.2 Request – Params

This API does not require any parameters within the request body.

Note: for TXPair connections, this API will return two records for each active TXPair connections, they will have same connection name and receiver name with different host. Each individual connection record contains stats data from each individual host connections.

2.10.8.3 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
    "active_connections": [
      {
        "connection_name": “[name of the associated KVM connection]”,
        "receiver_name": “[name of the receiver]”,
        "host": {
          “type”: “ConnectViaTx” | “VM” | “VMPool” | “VMHorizon” | “TXPair”,
          “value”: “[name of the target, either transmitter name or VM/VMPool IP]”
        },
        "active_user": “[name of the logged-in user]”,
        "zone": “[name of the zone connection is assigned to]”,  // if applicable
        "type": “Private” | “Shared”,
        "duration": “[active duration in seconds]”,
        "total_bandwidth": “[total average bandwidth in mbps]”,
        "video_bandwidth": “[average video bandwidth in mbps]”,
        "audio_bandwidth": “[average audio bandwidth in mbps]”,
        "usb_bandwidth": “[average USB bandwidth in mbps]”,
        "rtt": “[RTT in milliseconds]”,
        "user_latency": “[user latency in milliseconds]”,
        "fps": “[FPS]”,
        "dropped_fps": “[dropped FPS]”
      },
      ….  // Potentially multiple KVM active connections
    ]
  }
}
```

2.10.9 API for Getting KVM Active Connection Status Summary

This API is for obtaining a simplified set of attributes for active connections in Boxilla, in comparison with the detailed active connection status provided by the GET API defined in section 2.10.8. The difference between these two APIs also includes that, the attributes of any newly created active connections are accessible by calling this API instantly, while the API defined in section 2.10.8 needs to wait a maximum of 1 minute for the Boxilla service to update the newly created active connection information locally, together with the video/audio/usb traffic statistics.

The attributes returned by this API include the following contents:

- Connection name
- Receiver name
- Host information (group of connection, host IP)
- Name of logged-in user
- Connection type: private or shared

2.10.9.1 Request - Uri

Uri	bxa-api/connections/kvm/active/summary
Method	GET

2.10.9.2 Request – Params

This API does not require any parameters within the request body.

2.10.9.3 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
    "active_connections": [
      {
        "connection_name": “[name of the associated KVM connection]”,
        "receiver_name": “[name of the receiver]”,
        "host": {
          “type”: “ConnectViaTx” | “VM” | “VMPool” | “VMHorizon” | “TXPair”,
          “value”: “[name of the target, either transmitter name or VM/VMPool IP]”
        },
        "active_user": “[name of the logged-in user]”,
        "zone": “[name of the zone connection is assigned to]”,  // if applicable
        "type": “Private” | “Shared”,
        "view_only": “Yes” | “No”
      },
      ....  // Potentially multiple KVM active connections
    ]
  }
}
```

2.10.10 API for getting connection preset list

This API is for user to get a list of available preset defined in Boxilla DB, when username not provided, Boxilla return a full list of defined presets. An empty array will be returned if there is no preset defined, or there is no preset assigned to the user specified. The preset is ordered by the last modified date.

2.10.10.1 Request – Uri

Uri	bxa-api/connections/presets
Method	GET

2.10.10.2 Request – Params

Requirement	Name	Type	Description
Optional	username	String	Name of the KVM user trying to get preset list.

2.10.10.3 Request body JSON format

```
{
 "username": [username]
}
```

2.10.10.4 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-Version: v1

Body:
{
  “code”: “200”,
  “message”: {
    “presets”: [
       [List of defined preset names]
     ]
   }
}
```

Note: when there is no preset defined on Boxilla, or user doesn’t have permission to launch any Boxilla defined preset, the response message will be 200 with empty preset list.

2.10.10.5 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- “username” is invalid, for example, not a valid JSON string, or user not exist. The associated error message in the response body is as follows:
  "**message**": "Invalid user name."

2.10.11 API for getting preset details

This API is for getting the detailed information of connection preset.

2.10.11.1 *Request – Uri*

Uri	bxa-api/connections/presets/details
Method	GET

2.10.11.2 *Request – Params*

Requirement	**Name**	**Type**	**Description**
Optional	preset_names	String array	List of details of the connections for presets provided in the parameter. If the array is empty or parameter not provided, this API returns details for all the presets that accessible for user.
	username	String	Name of the KVM user trying to retrieve the preset details, if not provided, this API returns all the presets (or specified in preset_names parameter) defined in Boxilla.

2.10.11.3 *Request Body JSON format*

```
{
  "preset_names": [“[prest_1]”, “[preset_2]”, …],  // optional, can be empty array
  "username": [username]
}
```

2.10.11.4 *Response – Success*

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": {
    "presets":  [
      {
       "name": "[preset_name]",
       "type": "Full | Partial",
       "data":[ {
         "source": {
           "connection_name": "[connection_name]"
         },
         "destinations": [
            "[receiver name]",
             // more receiver names if shared connection
           ]
         }
        ]},

       {…} // more presets if exists
    ]
  }
}
```

2.10.11.5 *Response – Error: Bad Request*

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- “preset_names” parameter is not an array (for example, an integer is given instead) or any elements within are not string type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter preset_names: [preset _names], expecting a string array."
- One or more preset name is not found or invalid, the message body is as follows:
  "**message**": "Invalid preset name(s) or preset(s) not exist: [preset name]."
- “username” parameter is invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid username."

2.10.11.6 *Response – Error: UnAuthorized*

This error response may be caused by:

- One or more connections in the presets not assigned to user. The associated message in the response body is as follows:
  "**message**": "Operation is not authorized due to user *[username]* doesn’t have permission to check preset *[preset name]*."

2.10.12 API for getting preset status summary

This API is to query a specific preset “status”, particularly before user launches a preset, they can query this API to see if connections of this preset have been launched or not. We recommend user do this operation before or after activating any preset.

Note: Boxilla doesn't have a concept of "active preset", that means if all the connection within a preset has been launched in active connection by any reason, Boxilla considers the preset is "active".

2.10.12.1 Requets – Uri

Uri	bxa-api/connections/presets/active/summary
Method	GET

2.10.12.2 Request – Params

Requirement	Name	Type	Description
Mandatory	preset_name	String	Preset name that user wants to query for its status summary.
Optional	username	String	Name of the KVM user trying to retrieve the preset status summary.

2.10.12.3 Request Body JSON format

```
{
  "preset_name":  "Preset name to query",
  "username": [username]
}
```

2.10.12.4 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1

Body:

{
  "code": "200",
  "message": {
    "presets":  [
      {
       "name": "[preset_name]",
       "type": "Full | Partial",
       "data": {
         "source": {
           "connection_name": "[connection_name]"
         },
         "destinations": [
           {
            "receiver_name":  "[receiver name]",
            "status": "Online" | "Offline",
            "connection_status": "Active" | "Idle" | "Occupied",
            "launched_connection": "[connection_name]"
           },
           {…}// more receivers if shared connection
         ]
       }
      },

     {…} // more presets if exists
   ]
 }
```

The connection preset summary response is very similar to the detail, but the receivers returned also with the connection status, which shows if the source connection has been activated on the destination receiver. There could be three different status:

- Active: the source connection is currently active on the receiver
- Idle: the destination receiver is currently no active connection
- Occupied: the destination receiver is currently in active connection other than the source connection.
  In this status, the launched_connection also returns with the actual connection name that currently active on the receiver.

2.10.12.5 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required preset name is not provided. The associated error message in the response body is as follows:
  "**message**": "Preset name parameter is missing."
- The preset name is not found or invalid, the message body is as follows:
  "**message**": "Invalid preset name or preset not exists: [preset name]."
- "username" is invalid, for example, not a valid JSON string. The associated error message in the response body is as follows:
  "**message**": "Invalid username"

2.10.12.6 *Response – Error: UnAuthorized*

This error response may be caused by:

- The user credentials are correct but one or more connections in the preset not assigned to user. The associated message in the response body is as follows:
  "**message**": "Operation is not authorized due to user *[username]* doesn't have permission to check preset *[preset name]*."

2.10.13 API for launching preset

This API is for customer to launch a dedicated connection preset.

2.10.13.1 *Request – Uri*

Uri	bxa-api/connections/presets/activate
Method	POST

2.10.13.2 *Request – Params*

Requirement	Name	Type	Description
Mandatory	preset_name	String	Preset name to activate
	username	String	Name of the KVM user logged into the receiver.
	password	String	Password of the KVM user logged into the receiver.

2.10.13.3 *Request body JSON format*

```
{
  "preset_name":  "[Preset name]",
  "username":  "[username]",
  "password":  "[password]"
}
```

2.10.13.4 *Response – Success*

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
  "message": {
    "description": “The preset [preset name] has been activated successfully”
  }
 }
```

2.10.13.5 Response – Partial success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1
Body:
{
  "code": "200",
   "message": {
     "description": "The preset [preset name] has been activated, some receivers failed to launch",
     "error": {
        "failed_launches": [{
          "source": {
            "connection_name": "[connection_name]",
            },
          destinations": [
            "[receiver name]"
             // more receivers if shared connection
           ]
       ]
      }
   }
}
```

2.10.13.6 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required preset name / username / password is not provided. The associated error message in the response body is as follows:
  "**message**": "*[missing parameters]* parameter not provided."
- The preset name is not found or invalid, the message body is as follows:
  "**message**": "Invalid preset *[preset name]* or preset *[preset name]* doesn't exist."
- The preset contains a TXPair connection which not all device in the connection hosts are managed. The message body is as follows:

"**message**": "Not all trasmitters are managed."

- The preset contains a TXPair connection which not launching into a dual head receiver, the message body is as follows:
  "**message**": "Launching TX pair connection on single head device: *[aRX name]*."
- The preset contains more than one private connection from same transmitter trying to launch into multiple receivers, the error message body is as follows:
  "**message**": "Can not launch a private connection with multiple receivers *([connection name]*)"
- The preset contains one private connection and a shared/exclusive connection from same transmitter trying to launch into multiple receivers, the error message body is as follows:
  "**message**": "Can not launch a private connection with shared or exclusive connections (appliance IP *[ip address]*)"
- The preset contains more than one exclusive connection from same transmitter trying to launch into multiple receivers, the error message body is as follows:
  "**message**": "Can not launch more than one exclusive connections from same transmitter (*s*)"
- Other errors is as follows:
  "**message**": "[*error message attached]*"

2.10.13.7 Response – Error: UnAuthorized

This error response may be caused by:

- The username or password provided is incorrect. The associated message in the response body is as follows:
  "**message**": "Operation is not authorized due to invalid login credentials."
- The user credentials are correct but one or more connections in the preset not assigned to user. The associated message in the response body is as follows:

"**message**": "Operation is not authorized due to user *[username]* doesn't have permission to launch connection(s) *[connections]*."

2.11 APIs for Zone-oriented Operations

Resources on Boxilla (user-favorites, KVM receivers and connections) can be divided into multiple zones for further management, where each zone can be assigned with the resources based on customer requirement. The northbound REST API provides functionalities for zone management.

2.11.1 API for Getting Overview of all Zone Information

This API is for retrieving an overview of the information of all zones configured in Boxilla, including the following contents for each zone:

- Zone name
- Zone description

2.11.1.1 Request - Uri

Uri	bxa-api/zones/all
Method	GET

2.11.1.2 Request – Params

This API does not require any parameters within the request body.

2.11.1.3 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
 "code": "200",
 "message": {
  "zones": [
   {
    "name": "[name of the configured zone]",
    "description": "[description of the configured zone]"
   },
   ....  // Potentially multiple zones
  ]
 }
}
```

If there aren’t any zones configured in Boxilla, API returns the same success response with the “zones” field as an empty array.

2.11.2 API for Getting Information of Individual Zone

This API is for retrieving the following information of an individual zone configured in Boxilla:

- Zone name
- Zone description
- All Receiver devices assigned to the zone.
- All Connections assigned to the zone.
- All User favorites assigned to the zone.

2.11.2.1 Request - Uri

Uri	bxa-api/zones
Method	GET

2.11.2.2 Request – Params

Requirement	Name	Type	Description
Optional	name	String	Name of the zone.

If no parameter is provided by the API request (parameter is null), the operation would be considered as getting the detailed information of all zones and the response is returned accordingly.

2.11.2.3 Request Body JSON Format

```
{
  "name": "[name_of_the_zone]" // Optional, can be null
}
```

2.11.2.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  "code": "200",
  "message": {
    "zones_info": [
      {
        "name": "[name of the configured zone]",
        "description": "[description of the configured zone]",
        "user_favorites": [
          {
            "username": "[username in user favorite]",
            "favorite_settings": [
              {
                "favorite_number": "[user favorite number]",
                "connection_name": "[connection name in user favorite]"
              },
              …  // Potentially multiple settings for this user
            ]
          },
          …  // Potentially multiple user favorites
        ],
        "receivers": [
          {
            "name": "[receiver_name]"
          }
          …  // Potentially multiple receivers
        ],
        "connections": [
          {
            "name": "[connection_name]"
          }
          … // Potentially multiple connections
        ]
      },
      … // Potentially multiple zones
    ]
  }
}
```

If there aren't any zones configured in Boxilla, API returns the same success response with the "zone_infos" field as an empty array.

If no user favorites, receivers or connections are assigned to the specified zone, the associated fields within the response body ("user_favorites", "receivers" or "connections") are with values of empty arrays.

2.11.2.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- The zone specified by the "name" parameter does not exist. The associated error message in the response body is as follows:

"**message**": "Zone *[name]* does not exist."

2.11.3 API for Creating Zones

This API is for creating new zones with specific name and description attributes in Boxilla.

2.11.3.1 Request - Uri

Uri	bxa-api/zones
Method	POST

2.11.3.2 Request – Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the new zone. Maximum length 32.
	description	String	Description of the new zone. Maximum length 64.

2.11.3.3 Request Body JSON Format

```
{
  "name": "[name_of_the_zone]",
  "description": "[description of the zone]"
}
```

2.11.3.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "201",
  "message": "Successfully created zone [name]."
}
```

2.11.3.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Username/password or other required property parameters are passed in as empty or invalid. The associated error message in the response body is as follows:

"**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"

- The “name” string parameter includes characters that do not belong to the pre-defined character set (see section 4.3.3 for more details). The associated error message in the response body is as follows:
  "**message**": "Zone name invalid: *[name]*. Its characters must be from [0-9a-zA-Z.-] charset."

2.11.3.6 Response - Error: Name Parameter Length Exceeding Limit

This error response indicates that the name parameter within the request exceeds the length limit of 32.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": " Zone name length cannot exceed 32 characters."
}
```

2.11.3.7 Response - Error: Description Parameter Length Exceeding Limit

This error response indicates that the description parameter within the request exceeds the length limit of 64.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": " Zone description length cannot exceed 64 characters."
}
```

2.11.3.8 Response - Error: Zone Already Exists

This error response indicates that a zone with the specified name already exists and the API request is attempting to create a duplicate record.

```
HTTP/1.1 403 Forbidden

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  "code": "403",
  "message": "Duplicate zone name received: [name]. All zones must have a unique name."
}
```

2.11.4 API for Editing Name/Description of Individual Zone

This API is for editing name and description attributes of an existing zone in Boxilla.

2.11.4.1 Request - Uri

Uri	bxa-api/zones
Method	PUT

2.11.4.2 Request – Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the zone. Maximum length 32.
	new_name	String	New name of the zone to be edited. Maximum length 32.
	description	String	New description of the zone to be edited. Maximum length 64.

2.11.4.3 Request Body JSON Format

```
{
  "name": "[name of the zone]",
  "new_name": "[new name of the zone to be edited]",
  "description": "[description of the zone]"
}
```

2.11.4.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "200",
  "message": "Successfully updated zone [name]."
}
```

2.11.4.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Username/password or other required property parameters are passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- The "new_name" string parameter includes characters that do not belong to the pre-defined character set (see section 4.3.3 for more details). The associated error message in the response body is as follows:
  "**message**": "New zone name invalid: *[new_name]*. Its characters must be from [0-9a-zA-Z.-] charset."
- The zone specified by the "name" parameter does not exist. The associated error message in the response body is as follows:
  "**message**": "Zone *[name]* does not exist."

2.11.4.6 Response - Error: New_Name Parameter Length Exceeding Limit

This error response indicates that the new_name parameter (indicating the new name to be given to the specified zone) within the request exceeds the length limit of 32.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
```

```
{
  "code": "403",
  "message": " Zone name length cannot exceed 32 characters."
}
```

2.11.4.7 Response - Error: Description Parameter Length Exceeding Limit
This error response indicates that the description parameter within the request exceeds the length limit of 64.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": " Zone description length cannot exceed 64 characters."
}
```

2.11.4.8 Response - Error: Updating Zone Name with Existing Name
This error response indicates that the new name of the zone to be edited (specified by the new_name parameter value) is the same with the existing zone name.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Duplicate zone name received: [name]. All zones must have a unique name."
}
```

2.11.5 API for Editing Associations between Zones and KVM Receivers
This API is to edit the association between an existing zone and a KVM receiver which can be assigned to a zone.

2.11.5.1 Request - Uri

Uri	bxa-api/zones/devices/kvm/rx
Method	POST

2.11.5.2 Request - Params

Requirement	Name	Type	Description
Mandatory	zone_name	String	Name of the zone with the KVM receiver to be assigned to.
	receiver_names	String Array	List of names of the KVM receivers to be assigned to the zone. Can be an empty array.

2.11.5.3 Request Body JSON Format

{
"**zone_name**": *"[name of the zone]"*,
"**receiver _names**": *"[list of names of the receivers to be assigned to the zone]"*
}

2.11.5.4 Response - Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:
```

{
"**code**": "200",
"**message**": "Successfully assigned receivers *[receiver names]* to zone *[zone_name]*."
}

2.11.5.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as invalid or empty. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- No zone exists with the specified zone_name parameter in the request body. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone_name]* does not exist."

- One or more KVM receivers do not exist within the receiver_names array parameter in the request body. This could be caused by providing an invalid receiver name or a name pointing to a transmitter. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) do(es) not exist: *[list of receiver names not exist]*."
- One or more specified receivers are offline. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) is (are) offline: *[offline receiver names]*."
- Reassignment of one or more receivers fail due to the embedded software issue. The associated error message in the response body is as follows:
  "**message**": "The following receiver(s) is (are) currently not functinoal *[error_receiver_names*."

2.11.5.6 Response - Error: Zone Reassignment not Supported by Receiver Firmware

This error response indicates that the version of firmware deployed on the specified KVM receiver in request parameters does not support reassignment to zones. Usually this is caused by older versions of firmware without the associated appliance southbound REST API.

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "403",
  "message": "Zone reassignment is not supported by firmware type [receiver model] version
[firmware version] of the following receivers [list of unsupported receiver names]."
}
```

2.11.6 API for Editing Associations between Zones and KVM Connections

This API is to edit the association between an existing zone and a KVM connection which can be assigned to a zone.

2.11.6.1 Request - Uri

Uri	bxa-api/zones/connections/kvm
Method	POST

2.11.6.2 Request - Params

Requirement	Name	Type	Description
Mandatory	zone_name	String	Name of the zone to which the KVM connections to be assigned.
	connection_names	String Array	List of names of the KVM connections to be assigned to the zone. Can be an empty array.

2.11.6.3 Request Body JSON Format

{
 "**zone_name**": *“[name of the zone]”*,
 "**connection _names**": *“[list of names of the connections to be assigned to the zone]”*
}

2.11.6.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
```

{
 "**code**": "200",
 "**message**": "Successfully assigned connections *[connection_names]* to zone *[zone_name]*."
}

2.11.6.5 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Parameters are passed in as invalid or empty. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*. "
- No zone exists with the specified zone_name parameter in the request body. The associated error message in the response body is as follows:
  "**message**": "Zone *[zone_name]* does not exist."
- One or more KVM connections does not exist with the specified connection names parameter in the request body. The associated error message in the response body is as follows:

"**message**": "The following connection(s) do(es) not exist: *[list of non-exist connection names]*."

2.11.7 API for Deleting All Zones

This API is to delete one or more zones from Boxilla, specified by the request parameter which contains a list of existing names of zones to be deleted.

2.11.7.1 Request - Uri

Uri	bxa-api/zones/all
Method	DELETE

2.11.7.2 Request – Params

This API does not require any parameters within the request body.

2.11.7.3 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "200",
  "message": "Successfully deleted all zones."
}
```

2.11.7.4 Response – Error: Attempting to Delete Zones with Assigned Receivers

This error response indicates that one or more zones are assigned with receivers and cannot be deleted. The associated error message in the response body is as follows:

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "409",
  "message": " Fail to delete the following zone(s): [list of zones with assigned receivers]. The zones have been assigned to appliances."
}
```

2.11.8 API for Deleting Specific Zones

This API is to delete one or more zones in Boxilla, specified by the request parameter which contains a list of existing names of zones to be deleted.

2.11.8.1 Request - Uri

Uri	bxa-api/zones
Method	DELETE

2.11.8.2 Request - Params

Requirement	Name	Type	Description
Mandatory	zone_names	String array	List of names of the zones to be deleted. If array is empty with no zone names inside, the API request is successful but no zone would be deleted.

2.11.8.3 Request Body JSON Format

```
{
  "zone_names": [“[zone_1]”, “[zone_2]”, …]
}
```

2.11.8.4 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "200",
  "message": "Successfully deleted zone(s) [zone_names]."
}
```

2.11.8.5 Response – Success with Empty zone_names Parameter

```
HTTP/1.1 204 No Content
Accept: application/json;
```

```
Content-Type: application/json;
Accept-version: v1.1
```

2.11.8.6 Response - Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Required parameters are passed in as empty (null parameter, not parameter as an empty array) or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter(s): *[List of parameters detected as empty or invalid]*"
- “zone_names” parameter is not an array (for example, an integer is given instead) or any elements within are not String type. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter zone_names: *[zone_names]*, expecting a String array."
- One or more zones within the zone_name list parameter do not exist. The associated error message in the response body is as follows:
  "**message**": "The following zone(s) do(es) not exist: *[List of zone names do not exist]*"

2.11.8.7 Response – Error: Attempting to Delete Zones with Assigned Receivers

This error response indicates that one or more zones are assigned with receivers and cannot be deleted. The associated error message in the response body is as follows:

```
HTTP/1.1 409 Conflict
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "code": "409",
  "message": " Fail to delete the following zone(s): [list of zones with assigned receivers]. The zones have been assigned to appliances."
}
```

2.12 APIs for Remote App-oriented operations

2.12.1 API for Uploading Remote App Headless CLI Private Key

This API is to upload remote app headless CLI private key to Boxilla.

Uri	bxa-api/remoteapp/cli_data_key
Method	POST

2.12.1.1 Request - Params

Requirement	Name	Type	Description
Mandatory	private_key	File	Private key to be uploaded.

2.12.1.2 Request Body

```
form-data
  key: private_key
    value: file
```

2.12.1.3 Response - Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
 "code": "201",
  "status": "created",
  "data": {}
}
```

2.12.1.4 Response – Error: Attempting to upload private key with no file chosen

This error response indicates that Boxilla received empty private_key field (no file chosen to be uploaded).

```
HTTP/1.1 403 Forbidden
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
```

```
Body:

{
  "code": "403",
  "status": "Forbidden",
  "data": {
    "message": "Error, no key file specified"
  }
}
```

2.12.1.5 Response – Error: Attempting to upload invalid or corrupted private key

This error response indicates that Boxilla received invalid or corrupted private key. Possible cases:

- An attempt to upload public key instead of private
- Invalid header format
- Corrupted key

```
HTTP/1.1 403 Forbidden

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  "code": "403",
  "status": "Forbidden",
  "data": {
    "message": "Invalid or corrupted key"
  }
}
```

2.13 APIs for Retrieving Backup Files

2.13.1 API for Retrieving List of Backup Files

This API is to allow customers to retrieve a list of backup files from Boxilla.

2.13.1.1 Request - Uri

Uri	bxa-api/backups/list
Method	GET

2.13.1.2 Request – Params

This API does not require any parameter within the request body.

2.13.1.3 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/json;

Accept-version: v1.1

Body:

{
  “status”: “success”,
  “backups”: [{
   “filename”: “boxilla”,
   “datetime”: “08-11-2022 16:17:48”, // in dd-MM-yyyy HH:mm:ss format
   “size”: “13”,
    “version”:  “16-17-48”
  },
  {
    "filename": "boxilla",
     "datetime": "27-11-2022 02:00:12",
      "size": "1",
      "version": "02-00-12"
    }
]}
```

2.13.2 API for Downloading Backup Files from Boxilla

This API support to download specific version of the backup files from Boxilla.

2.13.2.1 Request – Uri

Uri	bxa-api/backups/download
Method	GET

2.13.2.2 Request – Params

The below parameters are optional, when user not giving any parameter, Boxilla will download the latest backup file if exists.

Requirement	Name	Type	Description
Conditionally Optional	Filename	String	The backup file name that user want to download, maximum length 100.

Conditionally Optional	Datetime	String	The date time for the backup file, in 'dd-MM-yyyy HH:mm:ss' format.
Conditionally Optional	Version	String	The version name for the backup file, usually this can get from the bxa-api/backups/list request, when user specify 'Latest' as the version name, the other two parameters are ignored, this will download the latest backup file from the system.

2.13.2.3 Response – Success

```
HTTP/1.1 200 OK
Accept: application/json;
Content-Type: application/bbx;
Accept-version: v1.1
Body: [backup binary file]
```

2.13.2.4 Response – Failed (missing params)

```
HTTP/1.1 400 Bad Request
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "status": "failed",
  "message": "One or more required params missing: filename, datetime"
}
```

2.13.2.5 Response – Failed (backup not exists)

```
HTTP/1.1 404 Not Found
Accept: application/json;
Content-Type: application/json;
Accept-version: v1.1
Body:
{
  "status": "failed",
   "message": "The backup file does not exists!"
}
```

2.14 APIs for Video Wall Oriented Operations

2.14.1 API for Getting List of Video Walls

2.14.2 Request - Uri

Uri	bxa-api/videowall/list
Method	GET

2.14.3 Request – Params

This API does not require any parameters within the request body.

2.14.4 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/bbx;

Accept-version: v1.1

Body:
{
  "code": 200,
  "message": {
    "videowalls": [
      {
        "id": 18,
        "name": "video wall 1",
        "description": "This is the video wall 1 description",
        "state": "Terminated"
      }
    ]
  }
}
```

2.15 API for Launching Video Wall

2.15.1 Request – Uri

Uri	bxa-api/videowall/activate
Method	POST

2.15.2 Request - Params

Requirement	Name	Type	Description

Mandatory	name	String	Name of the video wall.

2.15.3 Request Body JSON Format

```
{
   "name": "video wall 1"
}
```

2.15.4 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/bbx;

Accept-version: v1.1

Body:
{
   "code": 200,
   "message": "Video wall launched successfully",
   "session_id": "6b1e0f16bef75d459f6f"
}
```

2.15.5 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Video wall name property parameter is passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter, please provide valid video wall name."

2.16 API for Checking Video Wall Status

2.16.1 Request – Uri

Uri	bxa-api/videowall/active/status
Method	GET

2.16.2 Request – Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the video wall.

2.16.3 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/bbx;

Accept-version: v1.1

Body:
{
   "code": 200,
   "message": {
      "name": "video wall 3",
      "status": "Terminated",
      "session_id": ""
   }
}
```

2.16.4 Response – Error: Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Video wall name property parameter is passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter, please provide valid video wall name."

2.17 API for Terminating Video Wall

2.17.1 Request – Uri

Uri	bxa-api/videowall/terminate
Method	PUT

2.17.2 Request – Params

Requirement	Name	Type	Description
Mandatory	name	String	Name of the video wall.
	session_id	String	Session ID from the previous launch.

2.17.3 Request Body JSON Format

```
{
   "name": "video wall 3",
   "session_id": "6b1e0f16bef75d459f6f"
}
```

2.17.4 Response – Success

```
HTTP/1.1 200 OK

Accept: application/json;

Content-Type: application/bbx;

Accept-version: v1.1

Body:
{
   "code": 200,
   "message": "Video wall terminated successfully"
}
```

2.17.5 Request – Bad Request

This error response indicates any non-specific error (may be caused by internal errors within the API) returned, which might include:

- Video wall name property parameter is passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid parameter, please provide valid video wall name."
- Video wall session id property parameter is passed in as empty or invalid. The associated error message in the response body is as follows:
  "**message**": "Invalid session id."

3 REST Specific HTTP Error Status Codes

As described in section 2, the response messages of the APIs are divided into the success/error categories based on the HTTP status code. The 200 and 201 status codes are used in the success case and the rest of the codes are used in the error case, with a brief summary of the commonly-accepted usage as follows:

- **200 (OK)** – General success status code. Most common code to indicate success.
- **201 (CREATED)** – Successful creation occurred (via either POST or PUT). Set the Location header to contain a link to the newly-created resource. Response body content may or may not be present.
- **204 (NO CONTENT)** – Status when wrapped responses are not used and nothing is in the body (e.g. DELETE).
- **304 (NOT MODIFIED)** – Used in response to conditional GET calls to reduce band-width usage. If used, must set the Date, Content-Location, Etag headers to what they would have been on a regular GET call. There must be no response body.
- **400 (BAD REQUEST)** – General error when fulfilling the request would cause an invalid state such as domain validation errors, missing data, etc.
- **401 (UNAUTHORIZED)** – Error code for a missing or invalid authentication token.
- **403 (FORBIDDEN)** – Error code for user not authorized to perform the operation, doesn't have rights to access the resource, or the resource is unavailable for some reason (e.g. time constraints, etc.).
- **404 (NOT FOUND)** – Used when the requested resource is not found, whether it doesn't exist or if there was a 401 or 403 that, for security reasons, the service wants to mask.
- **409 (CONFLICT)** – Whenever a resource conflict would be caused by fulfilling the request. Duplicate entries, deleting root objects when cascade-delete not supported are a couple of examples.
- **500 (INTERNAL SERVER ERROR)** – The general catch-all error when the server-side throws an exception

4 Appendix

4.1 Appliance Property (Settings) Capabilities based on Device Class and Firmware Version

Supported Dev Type	Property	Valid Values	Supported Dev Class	Firmware
Transmitter	**Video Quality**	[“Best Quality”, “2”, “Default”, “4”, “Best Compression”]	Emerald-SE, Emerald-PE(single/dual-head); ZeroU	
	Video Source Optimization	[“Off”, “DVI/DP Optimised”, “VGA - High Performance”, “VGA - Optimised”, “VGA - Low Bandwidth”]	Emerald-SE, Emerald-PE(single-head); ZeroU	
	HID Configurations	0-5 Value representation: 0 - “Default” 1 - “Basic” 2 - “MAC” 3 - “Absolute” 4 - “Absolute MAC” 5 - “Dual Mouse”	Emerald-SE, Emerald-PE(single/dual-head); ZeroU	0-3: all firmware versions 4: >= 5.3.0 5: >= 5.4.1
			Emerald-4K	0-3: all firmware versions 4-5: >= 1.3.1
	Mouse_Keyboard_Timeout	0-5 (seconds)	Emerald-SE, Emerald-PE(single/dual-head); ZeroU	
			Emerald-4K	>= 1.4.0
	EDID Setting DVI 1	[“1920x1080”, “1920x1200”, “1680x1050”, “1280x1024”, “1024x768”, “Clone”]	Emerald-SE, Emerald-PE(single/dual-head); ZeroU	

		[“Default”, “3840x2160p-60Hz”, “3840x2160p-30Hz”, “2560x1440p-60Hz”, “1920x1080p-60Hz”, “Clone”]	Emerald-4K	>= 1.4.0
	EDID Setting DVI 2	[“1920x1080”, “1920x1200”, “1680x1050”, “1280x1024”, “1024x768”, “Clone”]	Emerald-SE, Emerald-PE(dual-head)	
Receiver	**HTTP-Enabled**	[“Enabled”, “Disabled”]	Emerald-SE, Emerald-PE(single/dual-head); Emerald-4K	

Table 2. KVM Appliance Property Capability based on Firmware

4.2 API Requests involving Multiple Operations

The following APIs involve potentially multiple operations on Boxilla objects, which are specified by the parameter as arrays:

Scope	**Request Type**	**URI**
Users	GET	bxa-api/users/kvm
	PUT	bxa-api/users/kvm/connections
	DELETE	bxa-api/users/kvm
		bxa-api/users/kvm/all
Devices	GET	bxa-api/devices/kvm
		bxa-api/devices/kvm/properties
		bxa-api/devices/kvm/status
	POST	bxa-api/devices/kvm/reboot
		bxa-api/devices/kvm/reboot-all
Connections	GET	bxa-api/connections/kvm
		bxa-api/connections/kvm/active
		bxa-api/connections/kvm/active/summary
	DELETE	bxa-api/connections/kvm
		bxa-api/connections/kvm/all

On failure of one or more individual operation, the corresponding API returns an error response with status code 400, 401, 403 or 409 accordingly, together with the response message listing each operation that fails. The other operations successfully executed are not affected. In this case, any external manager using the northbound REST API should send the GET request associated to the failed operations to confirm the non-reported operations (no error messages) were successful.

For example, when deleting a list of KVM connections using the API specified in section 2.9.4 and some specified connections did not get deleted due to existence of active connections, the API returns a HTTP 409 response with the message listing all the connections to be deleted that are currently active. When the API external manager receives the error response, it should send the GET request for connection status (API specified in section 2.9.5) to make sure that the other inactive connections specified in the parameter of the previous DELETE request have been successfully deleted.

4.3 Naming Validation Rules

Username/password credentials and names of Boxilla resources such as KVM device/connection names are used for multiple features included in the northbound REST APIs, which are validated at each request. This section briefly describes the naming rules used for validation by the APIs.

4.3.1 Rules for Valid Name of KVM Users/Devices/Connections

In general, names (including usernames/device_names/connection_names) should be set as easy-to-read, using ASCII characters to guarantee interoperability with external usage. The following sub-sections define some rules for name validation.

4.3.1.1 Allowed Characters

A valid name can contain only alphabetical characters (A-Z), numeric characters (0-9), and the following special characters:

- dash sign (-)
- underscore (_)
- period (.)
- parentheses (())
- white space (blank)

4.3.1.2 Disallowed Characters

- comma (,)
- tilde (~)

- colon (:)
- exclamation point (!)
- at sign (@)
- number sign (#)
- dollar sign ($)
- percent (%)
- caret (^)
- ampersand (&)
- apostrophe (')
- braces ({})

4.3.1.3 Length
The length of every name field should not be shorter than 2 characters and not longer than 64 characters.

4.3.1.4 Character Encoding
As Unicode characters are commonly supported in the Windows-based systems only, the use of Unicode characters should be avoided if requests via the northbound REST APIs use any non-Microsoft implementations.

It is recommended to use UTF-8 as the default encoding scheme.

4.3.2 Rules for Valid Password
Passwords should be set as easy-to-remember using the following rules:

4.3.2.1 Recommended Characters
Any password is recommended to contain the following contexts:

- At least one lower case letter
- At least one upper case letter
- At least one digit
- At least one of the special characters ~!@#$%^&*()_+

4.3.2.2 Length
The length of every password should not be shorter than 4 characters.

4.3.3 Rules for Valid Zone Name
Names of zones in Boxilla allow the following characters only:

- alphabetical characters (A-Z), both upper/lower cases
- numeric characters (0-9)
- dash sign (-)
- period (.)

Any other characters would be considered invalid for zone names and the associated northbound REST API would return error responses accordingly.

4.4 Recommended API Operation Work-flow Sequence to support: KVM User Login, Connection Launch, Connection Terminate and KVU User Logout

This section documents the recommended Northbound REST API command sequence from external controller entities to support the work flow sequence of User Login, Connection Launch, Connection Termination and User Logout. There are 8 steps in total, each command is followed by a Status Check Request to confirm the command executed successfully. The following is the recommended sequence:

1. KVM User Login Request.
2. Login Status Check Request.
3. KVM active connection Launch.
4. Launch Connection Status Check Request.
5. Terminate KVM active connection
6. Terminate Connection Status Check Request.
7. KVM User Logout Request.
8. Logout Status Check Request.

The sequence is described in details as the following steps:

1. REST client sends POST KVM user login API request (section 2.7.7).
2. On receiving login API response, REST client sends GET current KVM receiver status API request (section 2.8.6) and checks the following JSON field within the element matching the target KVM receiver in the API response:

   "**logged_in user**": {

     "**username**": *logged-in username*,

     "**type**": “active directory user” | “common user”

}
If the field is present in the response and the value of "username" matches the username of the logged KVM user, it is OK to proceed to the next step. Otherwise it is recommended to repeat this step until the expected username field is present in the response of this GET API. This ensures the KVM user login process on the receiver is complete before proceeding to the next step.

3. REST client sends POST launch-active-connection API request (section 2.9.6).
4. On receiving launching active connection API response, REST client sends GET active connection status summary API request (section 2.9.9) and checks whether the JSON element with the "connection_name" and "receiver_name" field values matching the expected active connection being launched and the target KVM receiver is present within the API response.
   If the element is present, it is OK to proceed to the next step. Otherwise it is recommended to repeat this step until the expected element is present in the response of the GET API. This ensures the active connection launching process on the receiver is complete before proceeding to the next step.
5. REST client sends DELETE terminate-active-connection API request (section 2.9.7).
6. On receiving terminating active connection API response, REST client sends GET active connection status summary API request (section 2.9.9) and checks whether the JSON element with the "connection_name" and "receiver_name" field values matching the expected active connection being terminated and the target KVM is still present within the API response.
   If the element is **not present anymore**, it is OK to proceed to the next step. Otherwise it is recommended to repeat this step until the expected element is removed from the response of the GET API. This ensures the active connection terminating process on the receiver is complete before proceeding to the next step.
7. REST client sends POST KVM user logout API request (section 2.7.8).
8. On receiving logout API response, REST client sends GET current KVM receiver status API request (section 2.8.6) and checks the "**logged_in_user**" JSON field within the element matching the target KVM receiver in the API response is **either empty or an alternative user**.
   If the field is empty or an alternative user, it is OK to proceed to the next step. Otherwise it is recommended to repeat this step until it's empty. This ensures the KVM user logout process on the receiver is complete before proceeding to any further API operations.

The recommended general operation sequence is separated to the following 4 parts accordingly:

- KVM user login (step 1) and login status check (step 2)
- Launch KVM active connection (step 3) and launching status check (step 4)

- Terminate KVM active connection (step 5) and terminating status check (step 6)
- KVM user logout (step 7) and logout status check (step 8)

Each operation status check present in step 2, 4, 6 and 8 would be a repeated process, with a 1-second time gap between each GET request. The check would be considered with a failure case if the response is not expected (for example, logged user is not included in device status response for login, or expected connection is not included in active connection summary request), and the repeated check would be terminated on either a successful check or up to 5 failed checks, before the REST client attempts to retry the corresponding request. Each request would experience maximally 3 retries in general, before claiming a final request failure.

The detailed sequence of KVM user login (step 1) and status check (step 2) is shown in Figure 1 as follows:

Northbound REST Request Sequence Step 1 & 2: KVM User Login and Status Check

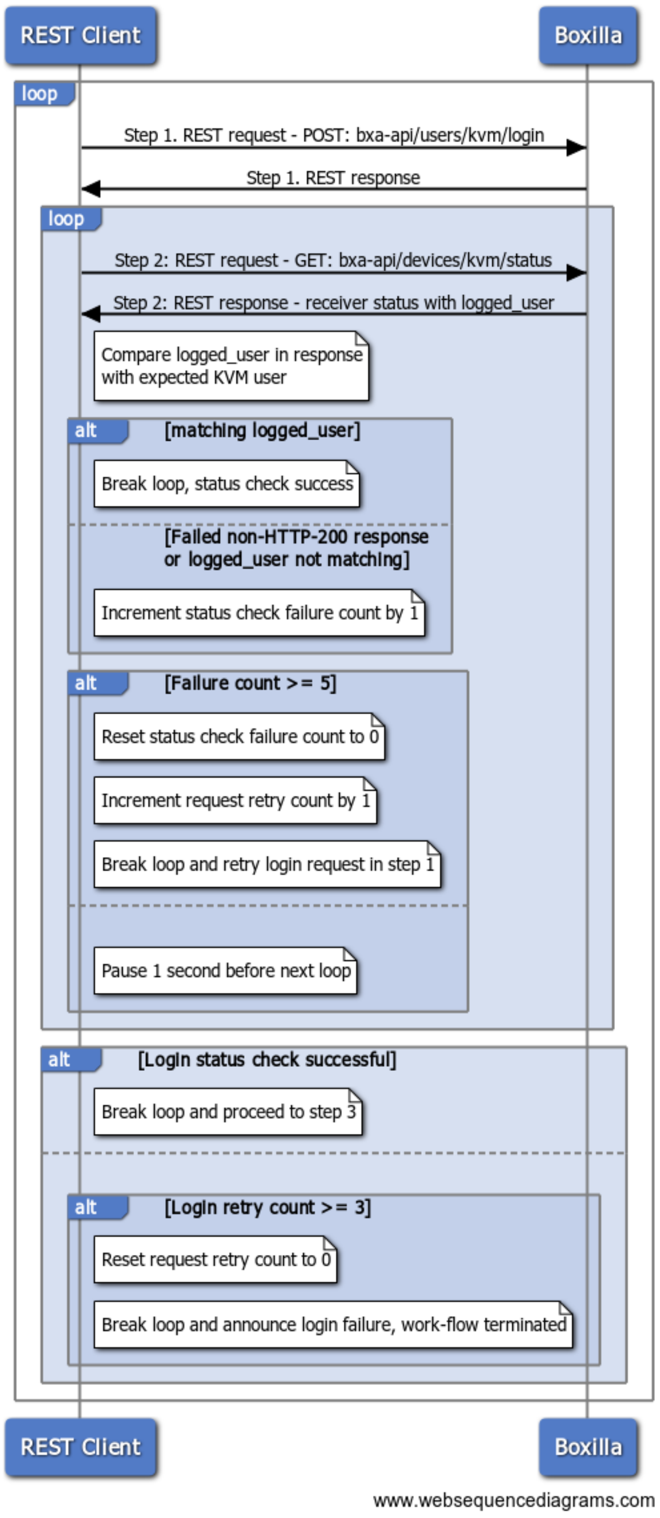

REST Client
Boxilla
loop
Step 1. REST request - POST: bxa-api/users/kvm/login
Step 1. REST response
loop
Step 2: REST request - GET: bxa-api/devices/kvm/status
Step 2: REST response - receiver status with logged_user
Compare logged_user in response with expected KVM user
alt
[matching logged_user]
Break loop, status check success
[Failed non-HTTP-200 response or logged_user not matching]
Increment status check failure count by 1
alt
[Failure count >= 5]
Reset status check failure count to 0
Increment request retry count by 1
Break loop and retry login request in step 1
Pause 1 second before next loop
alt
[Login status check successful]
Break loop and proceed to step 3
alt
[Login retry count >= 3]
Reset request retry count to 0
Break loop and announce login failure, work-flow terminated
REST Client
Boxilla
www.websequencediagrams.com

Fig. 1 Sequence of KVM user login request and status check (step 1 & 2)

The detailed sequence of launching KVM active connection (step 3) and status check (step 4) is shown in Figure 2 as follows:

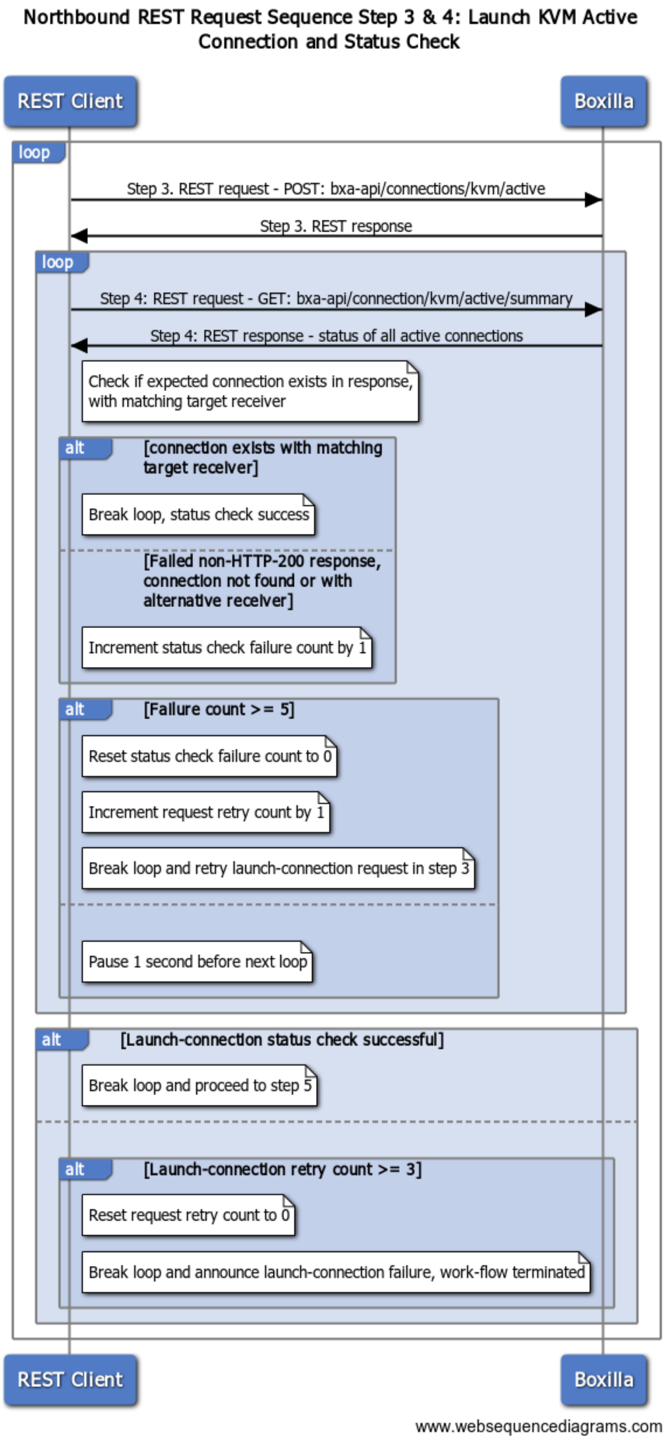

Northbound REST Request Sequence Step 3 & 4: Launch KVM Active Connection and Status Check
REST Client
Boxilla
loop
Step 3. REST request - POST: bxa-api/connections/kvm/active
Step 3. REST response
loop
Step 4: REST request - GET: bxa-api/connection/kvm/active/summary
Step 4: REST response - status of all active connections
Check if expected connection exists in response, with matching target receiver
alt
[connection exists with matching target receiver]
Break loop, status check success
[Failed non-HTTP-200 response, connection not found or with alternative receiver]
Increment status check failure count by 1
alt
[Failure count >= 5]
Reset status check failure count to 0
Increment request retry count by 1
Break loop and retry launch-connection request in step 3
Pause 1 second before next loop
alt
[Launch-connection status check successful]
Break loop and proceed to step 5
alt
[Launch-connection retry count >= 3]
Reset request retry count to 0
Break loop and announce launch-connection failure, work-flow terminated
REST Client
Boxilla
www.websequencediagrams.com

Fig. 2 Sequence of launching KVM active connection and status check (step 3 & 4)

The detailed sequence of terminating KVM active connection (step 5) and status check (step 6) is shown in Figure 3 as follows:

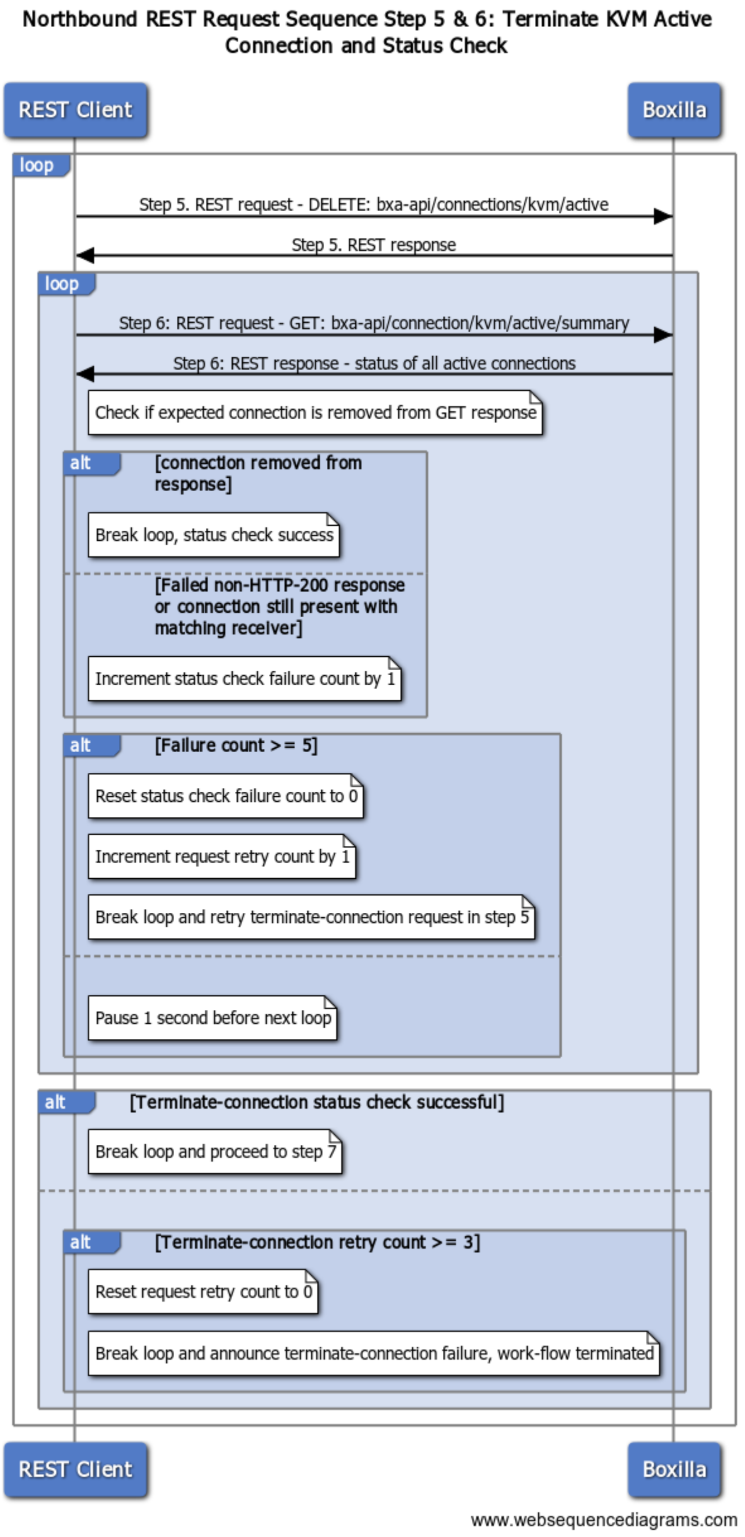
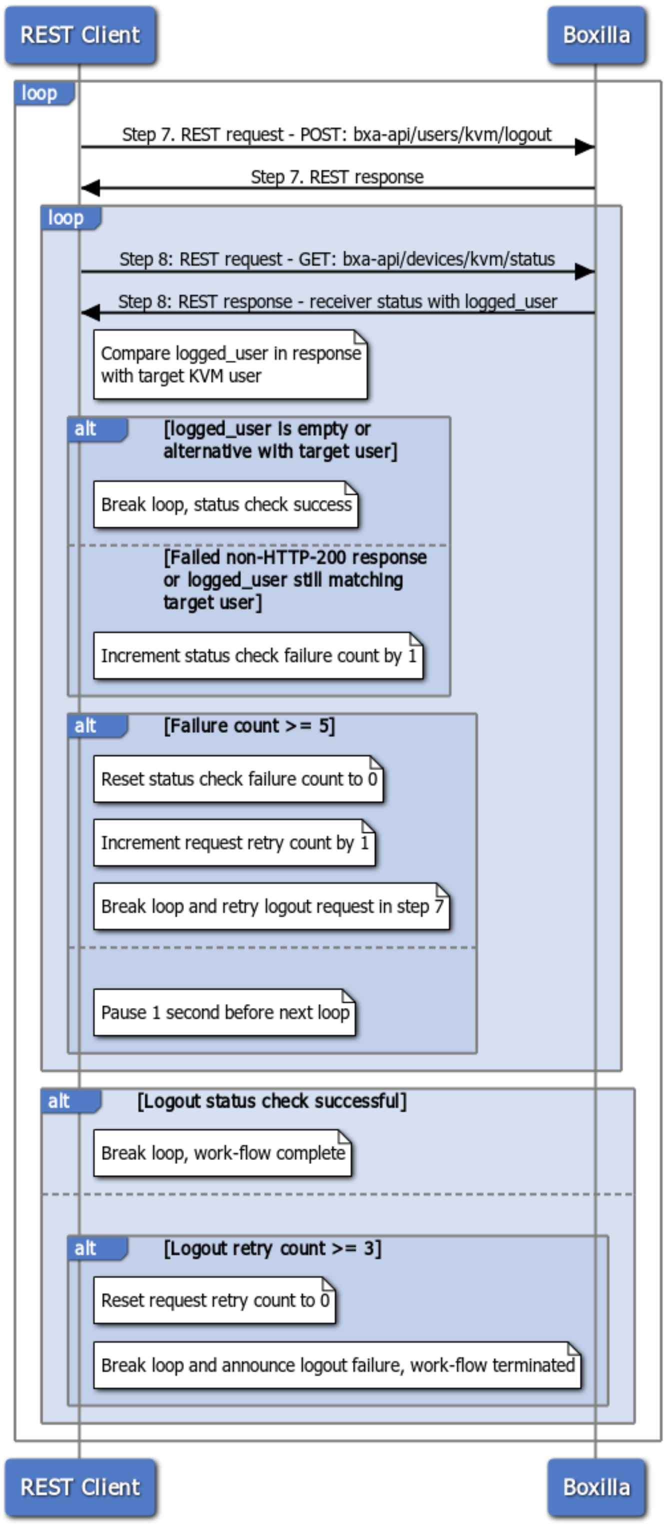

Northbound REST Request Sequence Step 5 & 6: Terminate KVM Active Connection and Status Check
REST Client
Boxilla
loop
Step 5. REST request - DELETE: bxa-api/connections/kvm/active
Step 5. REST response
loop
Step 6: REST request - GET: bxa-api/connection/kvm/active/summary
Step 6: REST response - status of all active connections
Check if expected connection is removed from GET response
alt
[connection removed from response]
Break loop, status check success
[Failed non-HTTP-200 response or connection still present with matching receiver]
Increment status check failure count by 1
alt
[Failure count >= 5]
Reset status check failure count to 0
Increment request retry count by 1
Break loop and retry terminate-connection request in step 5
Pause 1 second before next loop
alt
[Terminate-connection status check successful]
Break loop and proceed to step 7
alt
[Terminate-connection retry count >= 3]
Reset request retry count to 0
Break loop and announce terminate-connection failure, work-flow terminated
REST Client
Boxilla
www.websequencediagrams.com

Fig. 3 Sequence of terminating KVM active connection and status check (step 5 & 6)

The detailed sequence of KVM user logout (step 7) and status check (step 8) is shown in Figure 4 as follows:

Northbound REST Request Sequence Step 7 & 8: KVM User Logout and Status Check


REST Client
Boxilla
loop
Step 7. REST request - POST: bxa-api/users/kvm/logout
Step 7. REST response
loop
Step 8: REST request - GET: bxa-api/devices/kvm/status
Step 8: REST response - receiver status with logged_user
Compare logged_user in response with target KVM user
alt
[logged_user is empty or alternative with target user]
Break loop, status check success
[Failed non-HTTP-200 response or logged_user still matching target user]
Increment status check failure count by 1
alt
[Failure count >= 5]
Reset status check failure count to 0
Increment request retry count by 1
Break loop and retry logout request in step 7
Pause 1 second before next loop
alt
[Logout status check successful]
Break loop, work-flow complete
alt
[Logout retry count >= 3]
Reset request retry count to 0
Break loop and announce logout failure, work-flow terminated
REST Client
Boxilla

www.websequencediagrams.com

Fig. 4 Sequence of KVM user logout and status check (step 7 & 8)

4.5 Recommended API Operation work-flow sequence to support preset launch

4.5.1 Typical Work-flow for Launching a Preset From NBR

The typical operations for customer launching a preset over NBR are:

1. User sends get preset list request to check available presets
2. (optional) user sends get preset details to check the connection list for presets
3. (optional) User sends get preset status summary to check the current status of the preset
4. (conditional) user sends launch preset request to launch preset
5. (conditional) user sends get preset status summary request to check the status of the preset
6. (conditional) user sends get active connection summary request to cross check the preset launch result

The client software to launch prest over NBR can be broken down into two steps: retrieving preset information and launch (status check).

4.5.2 Retrieve Preset Information

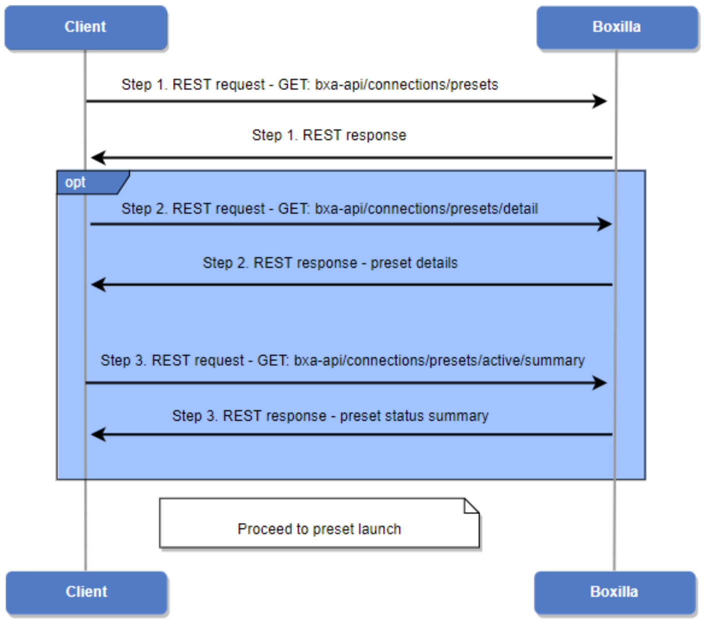

Client
Boxilla
Step 1. REST request - GET: bxa-api/connections/presets
Step 1. REST response
opt
Step 2. REST request - GET: bxa-api/connections/presets/detail
Step 2. REST response - preset details
Step 3. REST request - GET: bxa-api/connections/presets/active/summary
Step 3. REST response - preset status summary
Proceed to preset launch
Client
Boxilla

Figure 5 recommended workflow for retrieving preset information

The pre-check contains in total 3 steps:

1. Send get preset list request to get the available preset list.
2. Get preset details information.
3. Get preset status summary.

4.5.3 Launch Preset

4.5.3.1 *Launch Preset With 200 Success*

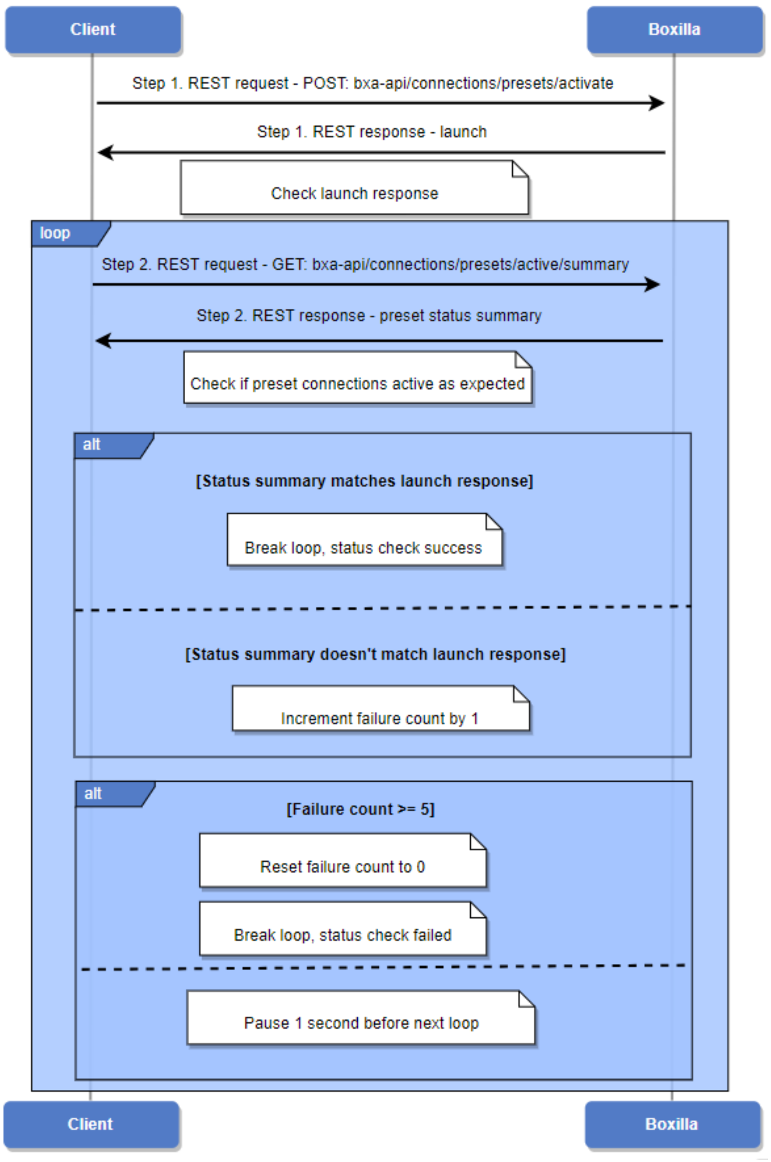

Client
Boxilla
Step 1. REST request - POST: bxa-api/connections/presets/activate
Step 1. REST response - launch
Check launch response
loop
Step 2. REST request - GET: bxa-api/connections/presets/active/summary
Step 2. REST response - preset status summary
Check if preset connections active as expected
alt
[Status summary matches launch response]
Break loop, status check success
[Status summary doesn't match launch response]
Increment failure count by 1
alt
[Failure count >= 5]
Reset failure count to 0
Break loop, status check failed
Pause 1 second before next loop
Client
Boxilla

Figure 6 recommended workflow for launching preset (200 success)

The launch and status check contains in total 2 steps:

1. Send the launch preset request to launch preset.

2. Send preset status summary request to check the connection status.

4.5.3.2 *Launch Preset With 200 Partial Success*

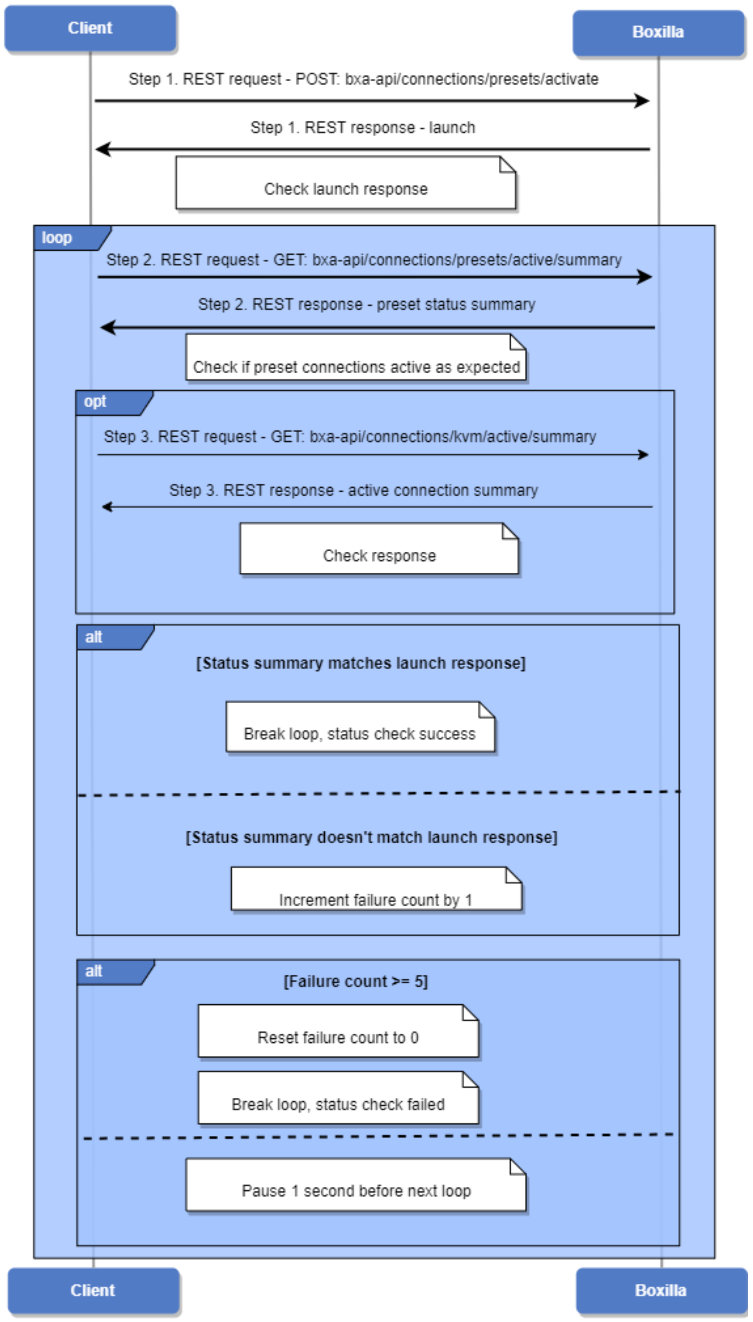

Client
Boxilla
Step 1. REST request - POST: bxa-api/connections/presets/activate
Step 1. REST response - launch
Check launch response
loop
Step 2. REST request - GET: bxa-api/connections/presets/active/summary
Step 2. REST response - preset status summary
Check if preset connections active as expected
opt
Step 3. REST request - GET: bxa-api/connections/kvm/active/summary
Step 3. REST response - active connection summary
Check response
alt
[Status summary matches launch response]
Break loop, status check success
[Status summary doesn't match launch response]
Increment failure count by 1
alt
[Failure count >= 5]
Reset failure count to 0
Break loop, status check failed
Pause 1 second before next loop
Client
Boxilla

Figure 7 recommended workflow for launching preset (200 partial success)

The launch and status check with 200 partial success have in total 3 steps:

1. Send the launch preset request to launch preset.
2. Send status summary request to check the connection status.
3. Cross check connection status with connections status summary API.

4.5.3.3 Re-launch Preset With Timeout

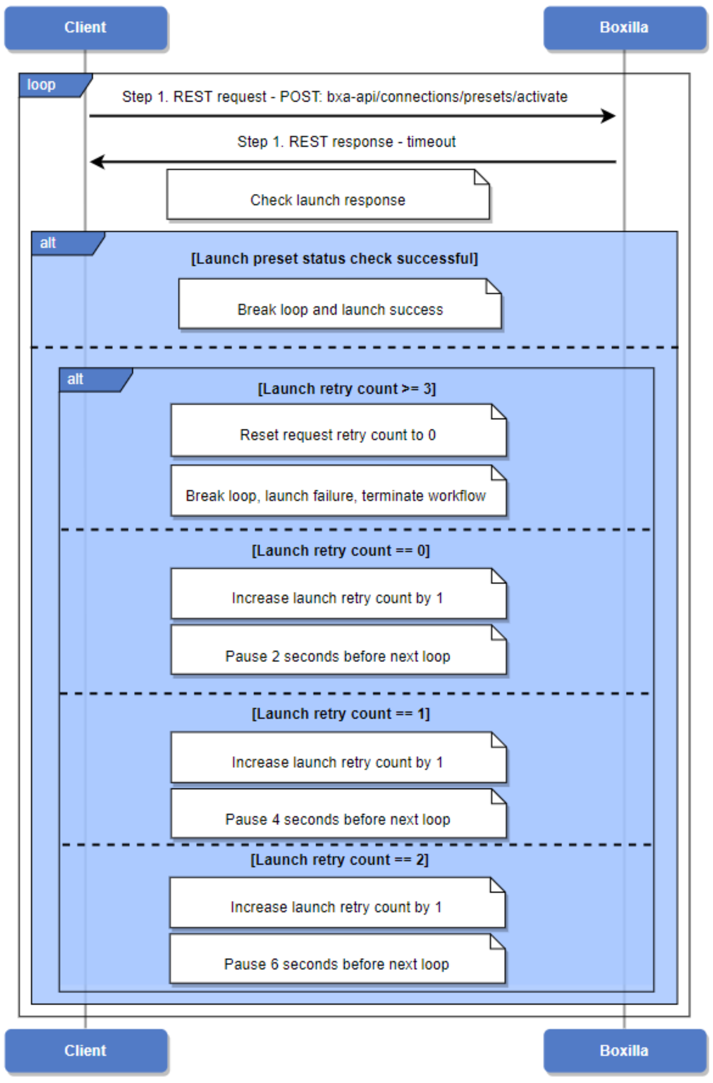

Client
Boxilla
loop
Step 1. REST request - POST: bxa-api/connections/presets/activate
Step 1. REST response - timeout
Check launch response
alt
[Launch preset status check successful]
Break loop and launch success
alt
[Launch retry count >= 3]
Reset request retry count to 0
Break loop, launch failure, terminate workflow
[Launch retry count == 0]
Increase launch retry count by 1
Pause 2 seconds before next loop
[Launch retry count == 1]
Increase launch retry count by 1
Pause 4 seconds before next loop
[Launch retry count == 2]
Increase launch retry count by 1
Pause 6 seconds before next loop
Client
Boxilla

Figure 8 recommended workflow for re-launch preset (timeout)

The recommended workflow for user software in the case that Boxilla failed to launch preset due to timeout:

1. Spawn a loop to handle retry launch if it fails for any reason.
2. Send the preset launch request to launch preset.
3. In the case it failed to launch the preset, pause for few seconds and retry to launch.
4. We recommend the user software has maximum retry limitation to prevent a dead lock.

4.5.3.4 Lauch Preset With 400 / 401 Error

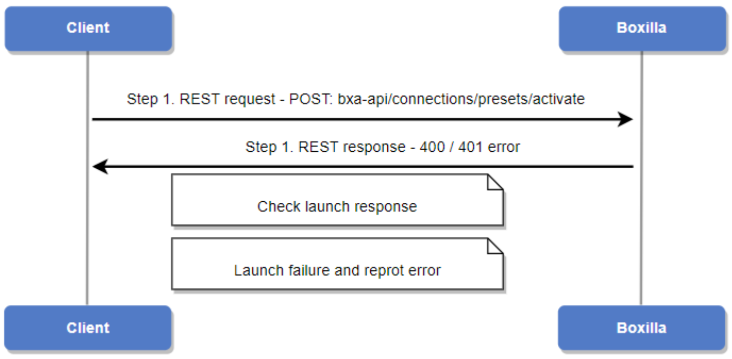

Client
Boxilla
Step 1. REST request - POST: bxa-api/connections/presets/activate
Step 1. REST response - 400 / 401 error
Check launch response
Launch failure and reprot error
Client
Boxilla

Figure 9 recommended workflow for re-launch preset (400 / 401 error)

The recommended workflow for launching preset while received 400 / 401 error:

1. Send launch request to launch preset.
2. Upon receive the 400 / 401 error, client should reject the launch and report error.