
AUTOMATION ANYWHERE ENTERPRISE

11.2

Control Room APIs - User Guide

Product Version	11.2.0.0
Document Version	1.0
Date of Publication	8/10/2018
Update(s) to Document Edition	Second Edition <ul style="list-style-type: none">▪ New Migration APIs - fetch delta since last migration▪ New API to deploy and monitor bot progress

Table of Contents

1. Control Room APIs	4
API for authentication	4
API to manage bot login credentials	4
API for bot deployment	4
API for export import of bots for Business Lifecycle Management	4
API to export and import Workload Management (WLM) configuration	4
API to get list of all queues	4
API to get all work items in a given queue	4
API to add/insert work item data to a given queue	5
API to manage the credential vault	5
API to manage data migration	5
API to deploy and monitor bot progress	5
2. Control Room API for Authentication	6
Authentication API	6
Authentication API Response Codes	6
3. Control Room API to manage bots login credentials	8
Authentication API	8
Login Credentials API	8
VB Script to Create User's Login Credentials	9
Contents of Input Files	10
4. Control Room API for Bot Deployment	11
Bot Deployment API Response Codes	12
5. API - Export and Import Bots for Bot Lifecycle Management	13
Features and benefits	13
Prerequisites	13
API Endpoints	13
Audit Logs	15
Import bots audit details	16
6. API - export and import workload management configuration	17
Features and benefits	17
Prerequisites	17
API Endpoints	17
Audit Logs	19
7. API - get all queues	20

API End Point.....	20
Get all queues.....	20
8. API - get all work items of a queue.....	22
API End Point.....	22
Get all queues.....	22
9. API - add/insert work item data to queue.....	24
API End Point.....	24
Steps to add work item data to a queue.....	24
10. APIs to manage credential vault.....	26
API end point.....	26
APIs to manage credentials.....	26
APIs to manage credential attributes.....	32
APIs to manage lockers.....	35
APIs to manage credential vault mode.....	42
API Response Codes.....	44
Audit Logs.....	45
11. API for data migration from 10.x Control Room to 11.x Control Room.....	46
API End Point.....	46
Migration Process APIs.....	46
Post Migration process APIs.....	55
A. Import files from My Docs folder.....	55
B. Migrate new or modified bots from 10.x since the last migration in 11.x.....	58
API Response Codes.....	59
Audit Logs.....	60
12. API for deploying and monitoring bot progress.....	61
API to fetch bot details.....	61
API to fetch list of available devices (bot runner clients).....	63
API to deploy bot.....	66
API to monitor bot execution.....	67
API Response Codes.....	69
Audit Logs.....	70

1. Control Room APIs

The Automation Anywhere Enterprise Control Room provides various public APIs which allow you to customize your business automation for third party applications. It enables the third party applications to consume RPA, orchestrate bots and manage the RPA data based on events.

Use the following APIs to manage your business automation using third party applications:

1. Authentication
2. Bot Login Credentials
3. Bot Deployment
4. Export Import Workload Management Configuration
5. Get list of all queues
6. Get list of all work items in a given queue
7. Insert work item data to a given queue
8. Credential Vault
9. Data Migration
10. Deploy and monitor bot progress

API for authentication

All the APIs are preceded by an Authentication API, wherein the user invoking third party application has to authenticate so as to use the downstream APIs. [Click here](#) for details.

API to manage bot login credentials

In enterprise organizations where the password rotation policy is applied, the Client user has to remember to update the password from Tools → Options → Login Settings during each password rotation. To automate this process, Control Room enables you to use an API to create, update or delete the Login Credentials stored in the credential vault. [Click here](#) for details.

API for bot deployment

To deploy Bots onto the Automation Environment the user has to login into Control Room, select the Bot and the Bot Runners and then 'Run/Schedule' the task. However, as the Automation scenarios scale up, there is an increasing need to deploy/trigger Bots from an external third party application. To meet this business requirement, you can use the Bot Deployment API which enables you to trigger a Bot from an External System. [Click here](#) for details.

API for export import of bots for Business Lifecycle Management

Usually, the Control Room user has to depend on means other than Control Room (for example email) to deploy TaskBots from one environment to another. Using the Export-Import APIs, you can easily introduce a customized BLM solution thus removing all external factors that could possibly disrupt your automation life cycle. [Click here](#) for details.

API to export and import Workload Management (WLM) configuration

Automation Anywhere has provided the Workload Management configuration API which can be used to export and import Workload configuration to move validated configurations from one environment to another. [Click here](#) for details.

API to get list of all queues

Automation Anywhere has provided a REST API that allows you to fetch the list and queue details available in the Control Room. [Click here](#) for details.

API to get all work items in a given queue

Automation Anywhere has provided a REST API that allows you to fetch the list of work items in a given queue and its details available in the Control Room. [Click here](#) for details.

API to add/insert work item data to a given queue

Automation Anywhere has provided a REST API that allows you to insert work item data in a given queue available in the Control Room. [Click here](#) for details.

API to manage the credential vault

Use the Credential Vault APIs to manage your Credentials, Attributes, Lockers, and Credential Vault mode in the Control Room. [Click here](#) for details.

API to manage data migration

Use the Migration APIs to manage data migration from source 10.x Control Room to destination 11.x Control Room database. [Click here](#) for details.

API to deploy and monitor bot progress

Use the Deploy and Monitor APIs to retrieve details of a given bot from the server repository to identify its file id to be used for bot deployment, fetch list of devices (bot runners) available for automation and its automation status, deploy a bot on given device(s) and fetch its automation id, and monitor the bot progress based on automation id. [Click here](#) for details

2. Control Room API for Authentication

Control Room v.11 exposes public APIs so as to manipulate the Control Room data and to deploy bots from an external system. This enables third party applications to consume RPA, orchestrate bots and to manage the RPA data based on events - for example, create an AAE user as soon as a user is created in SAP system; update the login credentials in AAE as soon as password is rotated in domain controller.

All the APIs are preceded by an Authentication API, wherein the invoking third party application user has to authenticate so as to use the downstream APIs.

Authentication API

API: v1/authentication

If the Control Room URL is <https://ultron.com:81>, then the API will be <https://ultron.com:81/v1/authentication>

The API takes two parameters as input in JSON format:

1. The username of the AAE user
2. The password of the AAE user

Note: All parameters are mandatory.

For example:

1. The AAE username is **mike_williams**.
2. The AAE password is **abc123**, then the JSON will be :

```
{
  "Username": "mike_williams",
  "Password": "abc123"
}
```

If the authentication is successful, the Control Room will issue an authentication token which needs to be passed on to the Deployment API as header information.

Note: The authentication token will ONLY be valid for 15 minutes from the time it is issued.

Authentication API Response Codes

Http Status code	Response	Description
200	<pre>{ "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyljoiMSIsImZcyI6Imh0dHA6Ly9sb2NhbGhvc3Qvd2ViY3JzdmMvliwiYXVkljoiaHR0cDovL2xvY2FsaG9zdC93ZWJjcnN2Yy8iLCJleHAiOiE0OTUwOTAwOTksIm5iZiI6MTQ5NTA4OTE5OX0.qPPhtj0j7LGAmWkj3XFymFfJXzA1P4zPehlJVYfulc" }</pre>	Authentication is successful
401	<pre>{ "message": "Invalid credentials." }</pre>	<ul style="list-style-type: none"> ◦ The password is invalid ◦ User does not exist ◦ AD Authentication - Credentials are Invalid
401	<pre>{ "message": "Please verify your email by clicking on the email verification link. This is mandatory as you will be able to login post verification only." }</pre>	Email notification is enabled - user has not verified email

	}	
402	{ "message": "License expired." }	License has expired
403	{ "message": "Your account is not activated. Please contact the admin." }	User is deactivated

3. Control Room API to manage bots login credentials

When the Bot is deployed from Control Room to the Bot Runner, the Bot will auto-login into the Bot Runner (if the machine is locked / logged off). The Bot will use the credentials stored in the Credential Vault for auto-login. These credentials are set by the user using the Tools → Options → [Login Settings](#) of AAE Client.

However, there could be cases when the user's Windows password is modified; especially in Enterprises where there is a password rotation policy. In such cases, the user has to remember to update the new password from Tools → Options → Login Settings.

In order to automate the above process, Control Room 11.1 provides a direct API to create, update or delete the Login Credentials stored in the credential vault.

There are 2 steps to use this API:

1. Invoke the Authentication API
2. Invoke the Login Credentials API

Authentication API

The details for the Authentication API are provided in the article [Control Room API for Authentication](#).

Login Credentials API

Note that only the Control Room Administrators (users having AAE_Admin Role) can use the Login Credentials API; this means that if the user invoking the Authentication API has an AAE_Admin role, only that user can use the Login Credentials API.

Also, this API will make use of the authentication token that is obtained using the Authentication API. The authentication token has to be passed on as one of the header inputs to the Login Credentials API.

API: v1/credentialvault/external/credentials/loginsetting

If the Control Room URL is https://ultron.com:81, then the API will be: https://ultron.com:81/v1/credentialvault/external/credentials/loginsetting

The API takes three parameters as input in JSON format:

1. The username of the AAE user.
2. The Login (Windows) username for the user which is to be updated in the Credential Vault against that user.
3. The Login (Windows) password for the user which is to be updated in the Credential Vault against that user.

Note: All parameters are mandatory.

For example:

If we take a 'Database Authentication' scenario (where users are stored and authenticated by the Control Room) and

1. The AAE username is **mike_williams**
2. The fully qualified Login username is **ultron.com\mike.williams**
3. The Login password is **abc123**, then the JSON will be

```
{ "Username": "mike_williams", "Windows_Username": "ultron.com\\mike.williams", "Windows_Password": "abc123" }
```

Using the Login Credentials API, the Control Room Admins can:

1. Create a user's Login Credentials in the Control Room - by using the 'POST' method
2. Update a user's Login Credentials in the Control Room - by using the 'PUT' method
3. Delete a user's Login Credentials from Control Room - by using the 'DELETE' method.

VB Script to Create User's Login Credentials

```
'AUTHENTICATION API - START
Set objStream = CreateObject("ADODB.Stream")
objStream.CharSet = "utf-8"
objStream.Open
objStream.LoadFromFile("D:\Deven.Deshpande\Office\Products\API for CV\auth-input.txt")

restRequest = objStream.ReadText()

objStream.Close
Set objStream = Nothing

contentType = "application/json"

Set oWinHttp = CreateObject("WinHttp.WinHttpRequest.5.1")
oWinHttp.Open "POST", "http://products1t12.aaspl-brd.com:81/v1/authentication", False
oWinHttp.setRequestHeader "Content-Type", contentType
oWinHttp.Send restRequest

response = oWinHttp.StatusText

MsgBox response

Dim AuthToken

'Set oJson = new aspJSON

AuthToken = oWinHttp.ResponseText

'oJson.loadJSON(AuthToken)

'MsgBox oJson.data("token")

MsgBox AuthToken

'AUTHENTICATION API - ENDS

'-----

'RESPONSE HEADER PARSING - START
Dim sToken
Dim posUser

sToken = Right(AuthToken, Len(AuthToken) - 10)

MsgBox(sToken)

'sToken = Left(sToken, Len(sToken) - 4)

posUser = InStr(sToken, "user")

MsgBox posUser

sToken = Left(sToken, posUser - 4)

MsgBox(sToken)

'RESPONSE HEADER PARSING - END

'DEPLOYMENT API - START

Set objStream_dep = CreateObject("ADODB.Stream")
objStream_dep.CharSet = "utf-8"
objStream_dep.Open
objStream_dep.LoadFromFile("D:\Deven.Deshpande\Office\Products\API for CV\cred-input.txt")

restRequest = objStream_dep.ReadText()
```

```
objStream_dep.Close
Set objStream = Nothing

contentType = "application/json"

Set oWinHttp = CreateObject("WinHttp.WinHttpRequest.5.1")

oWinHttp.Open "POST", "http://products1t12.aaspl-brd.-
com:81/v1/credentialvault/external/credentials/loginsetting", False

oWinHttp.setRequestHeader "Content-Type", contentType

oWinHttp.setRequestHeader "X-Authorization", sToken

oWinHttp.Send restRequest

response = oWinHttp.StatusText

MsgBox response

Dim DeployResponse

DeployResponse = oWinHttp.ResponseText

'MsgBox DeployResponse

'DEPLOYMENT API - ENDS
```

Contents of Input Files

Applicable to the above VB Script

The text Of auth-input.txt will have the input JSON String For authentication {"UserName":"admin","Password":"12345678"}

The text For deploy-input.txt will have the input JSON String For deployment

{ "Username": "mike_williams", "Windows_Username": "ultron.com\mike.williams", "Windows_Password": "abc123" }

4. Control Room API for Bot Deployment

To deploy Bots onto the Automation Environment, currently, the user has to login into Control Room, select the Bot and the Bot Runners and then 'Run/Schedule' the task.

However, as the Automation scenarios scale up, there is an increasing need to deploy/trigger Bots from an external third party application.

To meet this business requirement, Automation Anywhere Enterprise has published Application Programming Interfaces (APIs) using which a Bot can be triggered from an External System.

A Control Room user can use these APIs to deploy Bots (Tasks) to BotRunners on commencement of events specified by a third party/external application.

Key Features and Business Benefits of Control Room APIs

1. Bots can be deployed from an external third party systems using AAE APIs.
2. The input and output of APIs is JSON based (industry standard data-interchange format).
3. Bot Deployment can be orchestrated from an External Application / Workflow using a combination of scripts and AAE APIs.

Note: The Bot deployment API can ONLY be invoked once the system/user has authenticated using the [Authentication API](#)

Also, the user will need to have the 'Run my bots' privileges and the privileges of the bot runners on which the bot is to be deployed.

Deployment API

The Deployment API is used to deploy Bots to BotRunners.

API: <Control Room URL>/v1/schedule/automations/deploy

For example, if the Control Room URL is <https://www.ultron.com:81> ; then the bot deployment API will be <https://www.ultron.com:81/v1/schedule/automations/deploy>

The user can pass three parameters as JSON string.

1. Bot name with relative path - This is mandatory.
2. List of Bot-Runners and users in JSON format - This is mandatory.
3. Use RDP based approach - This is optional and set to **false** by default.

Deployment Scenario and corresponding JSON string:

1. For example, the name of the Bot is AccountsBot.atmx and the Bot is under 'My Tasks'
2. The Bot is to be deployed on 3 machines
 - First machine hostname BR-1 with user U-1
 - Second machine hostname BR-2 with user U-2
 - Third machine hostname BR-3 with user U3.
3. The JSON string in the above scenario will be:

```
{
  "taskRelativePath": "My Tasks\\AccountsBot.atmx", "botRunners":
  [
    {
      "client": "BR-1", "user": "U-1"
    },
    {
      "client": "BR-2", "user": "U-2"
    },
    {
      "client": "BR-3", "user": "U-3"
    }
  ]
}
```

Bot Deployment API Response Codes

Http(s) Status code	Response - Description
200	Successful creation of automaton.
400	Bad Request
401	Authentication Required
403	Unauthorized access
409	Conflict
500	Internal Server error

5. API - Export and Import Bots for Bot Lifecycle Management

Usually, the Control Room user has to depend on means other than Control Room (for example email) to deploy TaskBots from one environment to another. Using the Export-Import APIs, you can easily introduce a customized Bot Lifecycle Management (BLM) solution thus removing all external factors that could possibly disrupt your automation life cycle.

As a Control Room user with **Export bots** and **Download** bots permission, you can export a bot and its dependent files. Similarly, as a user with **Import bots** and **Upload** bots permission, you can import that bot and its dependent files.

For example, you can move the Bots that are verified as production ready from staging to production.

You can use the Control Room Export Import REST API to manage your automation TaskBots including dependent files in different environments such as Development, Testing, Acceptance, and Production based on your organization's automation needs.

Refer [Export bot files](#) and [Import bot files](#) to use the functionality from your Control Room user interface.

Features and benefits

1. Role based access control on Bot Lifecycle Management
2. Automatic export of dependencies (files and bots)
3. Audit and traceability on source and target environment for compliance
4. Email notification on successful execution or failure of export and import.

Prerequisites

Export

1. The Control Room user whose credentials are used for authentication must have **Export bots** permission
2. The Control Room user whose credentials are used for authentication must have **Download** permission on the Bots, minimum **Execute** permission on MetaBot, and dependencies that are being exported.
3. If Version Control is enabled in the source Control Room, the production version of all bots and dependencies which you want to export must be set.
4. User account that is used to run the Control Room services must have access to the location where package is getting exported, e.g. network location (shared drive) or on Control Room server machine

Import

1. The Control Room user whose credentials are used for authentication must have **Import bots** permission
2. The Control Room user whose credentials are used for authentication must have **Upload** permission on the Bots and dependencies that are being imported.
3. The Control Room user who will use the APIs to import multiple bots must have access to the exported package file provided by Automation Anywhere.
4. User account that is used to run the Control Room services must have access to the location where package is getting imported, e.g. network location (shared drive) or on Control Room server machine

API Endpoints

- a. **Export** - <Control Room URL>/v1/alm/export
For example, <https://crdevenv.com:81/v1/alm/export>
- b. **Import** - <Control Room URL>/v1/alm/import
For example, <https://crtestenv.com:82/v1/alm/import>

Using the above end points of the BLM Export Import API you can export and import a single bot and all of its dependencies.

Export Bot

Export a single bot with its dependent files using the Export API provided by Automation Anywhere:

1. Use the Post method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.
For example, <https://crdevenv.com:81/v1/authentication>
2. Use the Post method and state the parameters for credentials in Body Data. Refer sample:

```
{
  "username": "cradmin"
  "password": "cr@admin"
}
```

3. Click Play/Start

4. BLM Export API will make use of the authentication token that is obtained using the Authentication API. The authentication token has to be passed on as one of the header inputs to the BLM Export API.

5. Provide parameters such as **filename**, **destination path** and **package name** in Body Data to export a bot. Following list provides parameter description:

- **filename** - use this to specify the filepath ending with the name of the bot that you want to export
- **destination path** - use this to specify the destination path where the exported package is to be stored
- **package name** - use this to specify the package name that you want to assign to the exported package

Refer sample,

```
{
  "filePath": "Automation Anywhere\\My Tasks\\Finance\\Account Reconciliation.atmx",
  "destinationPath": "tempshare\\datashare\\Finance Department",
  "packageName": "Finance package"
}
```

6. Click Play/Start

7. You can use multiple sources to view the export results in,

- Reponse Data that comprise **package path** and **checksum**.
- Audit Log page (landing page as well as details page)
- Email when you receive notification on success or failure, if configured

Import Bot

Once the bot is successfully exported to a network drive or Control Room machine path, another authorized user can import that package to a different Control Room using the Import API:

1. Use the Post method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. Provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.
For example, **https://crtestenv.com:82/v1/authentication**
2. Use the Post method and state the parameters for credentials in Body Data.

Refer sample:

```
{
  "username": "cradmin2",
  "password": "cr@admin"
}
```

3. Click Play/Start

4. BLM Import API will make use of the authentication token that is obtained using the Authentication API. The authentication token has to be passed on as one of the header inputs to the BLM Import API.

5. Once your credentials are authenticated, provide parameters such as **package path** and the **checksum** that was generated as a token during export in Body Data.

Refer sample:

```
{
  "packagePath": "tempshare\\datashare\\Finance Department\\Finance Package_20171221-154403.aapkg",
  "checksum": "ZLyQ+Lbu2N+beEuXf6qd2Qi9uwi3BZxApn57C7mYjKQ="
}
```

Tip: You can copy the response of the **BLM Export API** and directly pass that as an input to the **BLM Import API**, if the package path is same and is accessible to the **BLM Import API** user.

API Response Codes


Http(s) Status code	Response - Description	Corrective Action
200	Package created successfully	NA
400	Bad request parameter	Retry with valid parameters
404	File not found	Ensure that the file/bot is present in Control Room
501	Permission error	Ensure that you have the Export/Import bots or Upload/Download permission

Audit Logs

An audit entry is logged in the Control Room **Audit Log** page when you export or import bots. The illustration below shows the detailed audit entries for Export Bots and Import Bots

Export bots audit details

Audit log > View action

 **Export bot files**
< Back

ACTION DETAILS

Status	Successful	Item name	Credit Card Bots Package_20180112-150708.aapkg
Action taken by	Mike	Time	2018-01-12 15:07:17 IST
Object type	Action	Action type	Export bot files
Source device	172.16.16.135	Source	API


EXPORT BOT FILES DETAILS

ATTRIBUTE	VALUE
Package File Path	\\datashare\Tempshare\Rajendra Vijay\BLM Demo\Credit Card Bot
Exported Bot	Automation Anywhere\My Tasks\Finance Bots\Credit Card Paymen
Exported Dependency(1)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process
Exported Dependency(2)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process
Exported Dependency(3)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process
Exported Dependency(4)	Automation Anywhere\My MetaBots\CyberArk Integration.mbot

Note: Cross import/export from API to User Interface (and vice versa) is not supported.

Import bots audit details

Audit log > View action

 **Import bot files**

< Back

ACTION DETAILS

Status

Successful

Action taken by

john

Object type

Action

Device

172.16.16.135

Item name

Credit Card Bots Package_20180112-150708.aapkg

Time

2018-01-12 16:04:21 IST

Action type

Import bot files

Source

API

IMPORT BOT FILES DETAILS

ATTRIBUTE	VALUE
Source Control Room	http://PRODUCTLT05.AASPL-BRD.COM:81/
Package File Path	\\datashare\Tempshare\Rajendra Vijay\BLM Demo\Credit Card Bot
Imported Bot	Automation Anywhere\My Tasks\Finance Bots\Credit Card Paymer
Imported Dependency(1)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process
Imported Dependency(2)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process
Imported Dependency(3)	Automation Anywhere\My Tasks\Finance Bots\Credit Card Process

6. API - export and import workload management configuration

As a Control Room administrator or a user with Export and Import bots permissions, you can export and import Workload configuration to move validated configurations from one environment to another.

For example, you can move the queues that are verified as production ready from staging to production.

You can use the Control Room Export Import REST API to manage your automation in different environments such as Development, Testing, Acceptance, and Production based on your organization's automation needs.

Features and benefits

- ▮ Role based access control on Workload Management module
- ▮ Audit and traceability on source and target environment for compliance

Prerequisites

Export

- ▮ The Control Room user whose credentials are used for authentication must be either **ADMIN** or should have **Export bots** permission
- ▮ The Control Room user must be either **Queue Admin** or should be **Owner** of the queue to export a particular queue
- ▮ Since details of queue Owners, Consumers, Participants are not part of the export, you need to manually add those as required after import.
- ▮ User account that is used to run the Control Room services must have read/write access permission to the folder where package is getting exported, i.e. the application path which could be either a network location (shared drive) or local machine.

For example,

C:\Users\Public\Documents\Server Files\wlm-files\export

- ▮ Maximum 200 queues can be exported in a package
- ▮ Maximum size of the package can be upto 10 GB (irrespective of number of files to export)
- ▮ Queues in draft state cannot be exported
- ▮ Entire export operation will fail, if any of the queue export fails.

Import

- ▮ The Control Room user whose credentials are used for authentication must have **Import bots** permission
- ▮ The Control Room Queue Admin can import **all** queues.
- ▮ The Control Room user who will use the APIs to import multiple bots must have **read /write** access permission to the folder where the exported package file shall be provided by Automation Anywhere. i.e. the application path which could be either a network location (shared drive) or local machine.

For example,

C:\Users\Public\Documents\Server Files\wlm-files\import

- ▮ The user who imports the package becomes the Owner of the imported queue by default. Add **Participants**, **Consumers** or additional **Owners** manually on the imported queues as required.
- ▮ If a queue category is already created by any previous import, subsequent imports will just use that queue category as long as all column names and types match.

API Endpoints

- a. **Export** - <Control Room URL>/v1/wlm/queues

For example,

https://crdevenv.com:81/v1/wlm/queues

- b. **Import** - <Control Room URL>/v1/wlm/queues

For example,

https://crtestenv.com:82/v1/wlm/queues

Using the above end points of the WLM Export Import API you can export and import queues.

Export Queues

To export queues using the Export API provided by Automation Anywhere:

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example, **https://crdevenv.com:81/v1/authentication**

2. Use the POST method and state the parameters for credentials in Body Data.

For example,

```
{
  "username": "Mike.Lee",
  "password": "1234567890"
}
```

3. Click Play/Start/Send
4. The Workload Export API will make use of the authentication token that is obtained using the Authentication API. The authentication token has to be passed on as one of the header inputs to the Workload Export API.
5. Provide Request parameters **X-Authorization**, **Accept** and **queue-ids** as Headers.

For example,

```
{
  "X-Authorization": "authorization token issued by Control Room after login",
  "Accept": "application/json/file",
  "queue-ids": "1,2,3,4,5"
}
```

6. Click Play/Start/Send
7. The action is successful when the response status is 200 OK.
8. Copy the package name from the response frame. You can use this to import the queue package.

Tip: Ensure the import package has the correct extension - **.w1mpkg**

For example, copy the **filename** given in **content-disposition**:

```
cache-control → no-cache, no-store, max-age=0, must-revalidate
content-disposition → attachment; filename=WorkloadPackage_20180524_180900.w1mpkg
content-length → 887
content-type → application/aapkg
date → Thu, 24 May 2018 12:39:00 GMT
expires → 0
pragma → no-cache
x-frame-options → SAMEORIGIN
```

Import Queues

Once the queue package is successfully exported to a network drive or Control Room machine path, another authorized user can import that package to a different Control Room using the Import API:

1. Use the Post method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. Provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.
For example, **https://crtestenv.com:82/v1/authentication**
2. Use the POST method and state the parameters for credentials in the request body.

Refer sample:

```
{
  "username": "Tom.Watson",
  "password": "1234567890"
}
```

3. Click Play/Start/Send
4. Workload Import API will make use of the authentication token that is obtained using the Authentication API. The authentication token has to be passed on as one of the header inputs to the Workload Import API.

5. Provide Request parameters **X-Authorization**, and **Content-Type** as Headers.

For example,

```
"X-Authorization": "authorization token issued by Control Room after login",
"Content-Type": "multipart/form-data"
```

6. Now provide the parameters **Content-Disposition: form-data; name="upload"; filename="{file name}"** to import the queue package in the request body.

For example,

```
Content-Disposition: form-data;
name="upload"; filename="{WorkloadPackage_20180524_180900.wlmpkg}"
```

7. Click Play/Start/Send

8. The action is successful when the response status is 200 OK.

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200	Package created successfully	NA
400	Bad request parameter	Retry with valid parameters
404	File not found	Ensure that the queue is present in Control Room
501	Permission error	Ensure that you have the required permission

Audit Logs

An audit entry is logged in the Control Room **Audit Log** page when you export or import queues. The illustration below shows the audit entries (successful as well as unsuccessful) for Export queues:

Audit log

Audit log

Action type ▼ Choose action type

Action type: Export queues ✕

Actions (9 of 112) 🔄 📄 🔍

<input type="checkbox"/>	STATUS	TIME	ACTION TYPE	ITEM NAME	ACTION TAKEN BY	SOURCE DEVICE	SOURCE	REQUEST ID
<input type="checkbox"/>	Successful	18:48:46 IST 2018-05-24	Export queues	WorkloadPackage_20180524_184846.wlmpkg	mike.lee	172.16.16.58	Control Room	ec66aaa7-ffc7-4d38-9...
<input type="checkbox"/>	Successful	18:09:00 IST 2018-05-24	Export queues	WorkloadPackage_20180524_180900.wlmpkg	mike.lee	172.16.16.58	Control Room	f99e6daa-3990-4d83-...
<input type="checkbox"/>	Unsuccessful	18:07:53 IST 2018-05-24	Export queues	--	ellie.brown	172.16.16.58	Control Room	b49b5db2-9475-47b1-...
<input type="checkbox"/>	Successful	16:27:55 IST 2018-05-24	Export queues	WorkloadPackage_20180524_162755.wlmpkg	tom.watson	172.16.16.58	Control Room	a8705960-b57b-4ee2-...
<input type="checkbox"/>	Successful	16:27:32 IST 2018-05-24	Export queues	WorkloadPackage_20180524_162732.wlmpkg	tom.watson	172.16.16.58	Control Room	9a5833c8-c146-4e34-...
<input type="checkbox"/>	Unsuccessful	16:25:23 IST 2018-05-24	Export queues	--	tom.watson	172.16.16.58	Control Room	f650b8d5-2249-4167-...
<input type="checkbox"/>	Unsuccessful	16:22:37 IST 2018-05-24	Export queues	--	tom.watson	172.16.16.58	Control Room	bcfb42c2-fd19-4029-8...
<input type="checkbox"/>	Unsuccessful	16:21:17 IST 2018-05-24	Export queues	--	tom.watson	172.16.16.58	Control Room	29833f15-b6e6-4d25-a...
<input type="checkbox"/>	Unsuccessful	16:21:05 IST 2018-05-24	Export queues	--	tom.watson	172.16.16.58	Control Room	86fa4dcb-c7d0-45ce-...

7. API - get all queues

As a Control Room user with Export and Import Bots and Queue Admin role permissions, you can use the Get all queues API fetch the list of work items of **all queues** and its details available in the Control Room.

Similarly as Control Room **user** with **Export** and **Import Bots** permissions as well as **Queue Owner**, **Participant**, and/or **Consumer** privileges, you can use the Get all queues API to fetch the list of work items **in a given queue** and its details available in the Control Room.

API End Point

Use the following end point to access the API:

<Control Room URL>/v1/wlm/queues

For example,

https://crdevenv.com:81/v1/wlm/queues

Get all queues

1. Use the Post method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

https://crdevenv.com:81/v1/authentication

2. Use the GET method to access the Workload APIs

For example,

GET https://crdevenv.com:81/v1/wlm/queues

3. Provide the following Request parameters in Headers

```
"X-Authorization" : "Authorization token"
"Content-Type" : "application/json"
```

4. Click **Send**

5. The action is successful when the response status is 200 OK.

6. You can view the result in the Body data.

For example,

```
[
{
  "id": 1, "createdBy": 2, "createdOn": "2018-05-23T16:53:46+05:30", "updatedBy": 2, "updatedOn":
"2018-05-24T16:26:02+05:30", "tenantId": 1, "version": 2, "name": "Invoice Queue", "description": "",
  "reactivationThreshold": 1, "status": "IN_USE", "expiry": null, "manualProcessingTime": null, "manu-
alProcessingTimeUnit": "SECONDS", "workItemProcessingOrders": [ { "id": 3, "priority": 1, "columnId":
2, "sortDirection": "asc" } ], "displayColumnIds": null, "workItemModelId": 1
},
{
  "id": 2, "createdBy": 2, "createdOn": "2018-05-25T12:28:36+05:30", "updatedBy": 2, "updatedOn":
"2018-05-25T12:28:36+05:30", "tenantId": 1, "version": 0, "name": "Devices for deployment queue",
  "description": "", "reactivationThreshold": 1, "status": "NOT_IN_USE", "expiry": null, "manu-
alProcessingTime": null, "manualProcessingTimeUnit": "SECONDS", "workItemProcessingOrders": [ { "id":
4, "priority": 1, "columnId": 6, "sortDirection": "asc" } ], "displayColumnIds": null,
  "workItemModelId": 2
}
]
```

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200	Queue details listed successfully	NA
400	Bad request parameter	Retry with valid parameters
404	File not found	Ensure that the queue is present in Control Room
501	Permission error	Ensure that you have the required permission

8. API - get all work items of a queue

As a Control Room user with Queue Admin role or **Queue Owner**, **Participant**, and/or **Consumer** privileges, you can use the Get all queues API to access all work items of a particular queue.

API End Point

Use the following end point to access the API:

<Control Room URL>/v1/wlm/queues/<queue-id>/workitems

For example,

https://crdevenv.com:81/v1/wlm/queues/1/workitems

Get all queues

1. Use the Post method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

https://crdevenv.com:81/v1/authentication

2. Use the GET method to access the Workload API and the work items within a specific queue

For example,

GET https://crdevenv.com:81/v1/wlm/queues/1/workitems

3. Provide the following Request parameters in Headers

"X-Authorization" : "Authorization Token"

"Content-Type": "application/json"

4. Click **Send**

5. The action is successful when the response status is 200 OK.

6. You can view the result in the Body data.

For example,

```
[
  { "id": 1, "createdBy": 2, "createdOn": "2018-05-23T16:54:15+05:30", "updatedBy": 2, "updatedOn": "2018-05-23T16:54:15+05:30", "tenantId": 1, "version": 0, "deviceId": null, "status": "ERROR", "startTime": null, "endTime": null, "brUserId": null, "queueId": 1, "comment": null, "botId": null, "poolId": null, "automationId": null, "data": { "Invoice Id": "INV4001", "Customer Name": "Mymm", "Amount": "", "email": "bfreshwater0@nps.gov", "Invoice Date": "2018-04-17T02:33:32Z" }, "lastPausedTime": null, "totalPausedTime": 0 },
  { "id": 2, "createdBy": 2, "createdOn": "2018-05-23T16:54:15+05:30", "updatedBy": 2, "updatedOn": "2018-05-23T16:54:15+05:30", "tenantId": 1, "version": 1, "deviceId": null, "status": "NEW", "startTime": null, "endTime": null, "brUserId": null, "queueId": 1, "comment": null, "botId": null, "poolId": null, "automationId": null, "data": { "Invoice Id": "INV4002", "Customer Name": "Vipe", "Amount": 1025, "email": "ibricksey1@godaddy.com", "Invoice Date": "2018-04-14T08:52:35Z" }, "lastPausedTime": null, "totalPausedTime": 0 },
  { "id": 3, "createdBy": 2, "createdOn": "2018-05-23T16:54:15+05:30", "updatedBy": 2, "updatedOn": "2018-05-23T16:54:15+05:30", "tenantId": 1, "version": 1, "deviceId": null, "status": "NEW", "startTime": null, "endTime": null, "brUserId": null, "queueId": 1, "comment": null, "botId": null, "poolId": null, "automationId": null, "data": { "Invoice Id": "INV4003", "Customer Name": "Skyndu", "Amount": 1466, "email": "vwooster2@nymag.com", "Invoice Date": "2018-04-25T14:02:29Z" }, "lastPausedTime": null, "totalPausedTime": 0 },
  ....]
```

API Response Codes

Http(s) Status code	Response - Descrip-	Corrective Action
---------------------	---------------------	-------------------

	tion	
200	Work items listed successfully	NA
400	Bad request parameter	Retry with valid parameters
404	File not found	Ensure that the queue is present in Control Room
501	Permission error	Ensure that you have the required permission

9. API - add/insert work item data to queue

As a Control Room administrator or a user with **Export** and **Import Bots** and **Queue Admin** role permissions as well as **Queue Owner**, **Participant**, and/or **Consumer** privileges, you can use the REST API provided by Automation Anywhere to add work item data to queues created in the Control Room.

API End Point

Use the following end point to access the API:

<Control Room URL>/v1/wlm/queues/<queue-id>/workitems

For example,

https://crdevenv.com:81/v1/wlm/queues/1/workitems

Steps to add work item data to a queue

1. Login to Control Room. Refer [Log on to Control Room](#) for more information.
2. Create a queue. Refer [Create a queue](#) for more information
3. For the next step, you need to note down the Queue Id from the database that you configured for your Control Room. Refer AAE Control Room Installation Guide for more information.

For example,

- a. Go to Control Room Database
 - b. Right click dbo.QUEUES in the Tables node
 - c. Click Select Top 1000 Rows to run the SQL query for Queues.
 - d. Identify the queue **id** that is displayed beside the queue **name** column - in this case **1**
4. Use POST method to the insert work item data to queue API to upload work items.

For example, if the Queue Id is '1', use the following parameters,

https://crdevenv.com:81/v1/wlm/queues/1/workitems

5. Provide the following Request parameters in Headers

"X-Authorization" : "Authorization token"

"Content-Type" : "application/json"

6. In the Request Body provide required parameters i.e. the work item column names with its data to insert it in a queue. Ensure the parameters for columns in the body data match that of the queue structure.

For example,

```
[{"Invoice Id":"INV5001","Customer Name":"Daniel","Amount":"55555","email":"daniel@aae.com","Invoice Date":"2018-05-28"}]
```

7. Click **Send**
8. The action is successful when the response status is 200 OK.
9. You can view the result in the Body data.

For example,

```
[
{
"id": 11, "createdBy": 2, "createdOn": "2018-05-28T12:29:22+05:30", "updatedBy": 2, "updatedOn": "2018-05-28T12:29:22+05:30", "tenantId": 1, "version": 0, "json": "q+j74omkZEMS0Ta2obN0Pvm-b1se/xK+MDc3k0/V4TOXfm1ICoG2jNEsT/yPlW5Xc64opOqAuozVoSf8Hg/us3HqIOtKRun-IJB4is/rCG9xK7g9LN+e3Cg4DTlgfoxD9UeB0oU+bXKKqlb3p6EI9f07gvvrymM45pIsoktSkM3+U=", "deviceId": null, "status": "NEW", "startTime": null, "endTime": null, "col1": "INV5001", "col2": "Daniel", "col3": "55555", "col4": "daniel@aae.com", "col5": "2018-05-28", "brUserId": null, "queueId": null, "comment": null, "botId": null, "poolId": null, "automationId": null, "lastPausedTime": null, "totalPausedTime": 0
```



```
}  
]
```

10. Go to Control Room → Workload → Queues → View page to verify that the data is added:

Workload > Queues > View queue

Invoice Queue

Edit... Delete... < Back

QUEUE DETAILS

Name
Invoice Queue

Description
--

My access
Queue owner

Status
Not in use

WORK ITEMS

GENERAL

OWNERS

PARTICIPANTS

CONSUMERS

WORK ITEM STRUCTURE

Work items

Automation name
N/A

Bot name
N/A

Device pool
N/A

Status
Choose status

Work items (11 of 11)

<input type="checkbox"/>	ID	STATUS	INVOICE ID	CUSTOMER NAME	AMOUNT	EMAIL	INVOICE DATE	START TIME	END TIME	MODIFIED BY
<input type="checkbox"/>	11	Ready to run	INV5001	Daniel	55555	daniel@aae.co...	00:00:00 IST 2018-05-28	N/A	N/A	mike.lee
<input type="checkbox"/>	10	Ready to run	INV4010	Voomm	305	elambal9@en...	04:44:18 IST 2018-05-05	N/A	N/A	mike.lee
<input type="checkbox"/>	9	Ready to run	INV4009	Photojam	1084	agarnsey8@ih...	09:09:27 IST 2018-05-08	N/A	N/A	mike.lee
<input type="checkbox"/>	8	Ready to run	INV4008	Ntags	604	cconnold7@ec...	06:36:33 IST 2018-05-12	N/A	N/A	mike.lee
<input type="checkbox"/>	7	Ready to run	INV4007	Wikivu	1460	fdrynan6@um...	12:12:22 IST 2018-04-18	N/A	N/A	mike.lee
<input type="checkbox"/>	6	Data error	INV4006	Feedspan	--	ymanuud5@...	22:02:41 IST 2018-04-16	N/A	N/A	mike.lee
<input type="checkbox"/>	5	Ready to run	INV4005	Flashset	1173	bpetrosian4@j...	11:34:13 IST 2018-04-19	N/A	N/A	mike.lee

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200	Work item data inserted successfully	NA
400	Bad request parameter	Retry with valid parameters
404	File not found	Ensure that the queue is present in Control Room
501	Permission error	Ensure that you have the required permission

Copyright © 2018 Automation Anywhere, Inc.

25

https://support.automationanywhere.com

10. APIs to manage credential vault

As a Control Room user with **Manage my credentials and lockers** role permissions, use the Credential Vault APIs to manage your Credentials, Attributes, Lockers, and Credential Vault mode in the Control Room.

1. The [Credential APIs](#) can be used to create, update, and delete credentials
2. The [Credential Attribute APIs](#) can be used to create, update, and delete the credential attributes and values.
3. The [Locker APIs](#) can be used to add, edit, and remove lockers including consumers and members
4. The [Credential Vault mode APIs](#) can be used to configure the **Connection mode** that allows you to connect to the Credential Vault using a **Master key**.

Note: The examples provided in this article are for reference only.

API end point

Use the following end point to access the Credential Vault APIs:

<Control Room URL>/v2/credentialvault/credentials

APIs to manage credentials

Use these APIs to manage your Credentials in the Control Room.

Before accessing the Credential Vault API's you must first use the authentication API and pass it as a token to use a particular Credential Vault API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

`https://crdevenv.com:81/v1/authentication`

2. Provide the following request payload in Headers
 "X-Authorization" : "Authorization token"
 "Content-Type" : "application/json"
3. Provide the following request payload in Body:


```
{
  "username": "<Username>",
  "password": "<Password>"
}
```

For example,

```
{
  "username": "Ellie.Brown",
  "password": "12345678"
}
```

1. Create Credential

This API allows you to create a credential.

<Control Room URL>/v2/credentialvault/credentials

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to provide the following request payload in Body:

```
{
  "name": "<Credential Name>",
  "description": "<Credential Description>",
  "attributes":
  [
    {
      "name": "<Attribute Name>",
      "description": "<Attribute Description>",
      "userProvided": <true or false>,

```

```
"masked": <true or false>
}
]
}
```

- **For example**, the following creates a Credential **"SharePoint Login"** with **"Username"** as an attribute:

```
{
  "name": "SharePoint Login",
  "description": "Credentials to login to SharePoint",
  "attributes":
  [
    {
      "name": "Username",
      "description": "Name of the user",
      "userProvided": true,
      "masked": true
    }
  ]
}
```

3. Click **Send**

4. The action is successful when the response status is 201 **Credential created**

5. You can view the response in the Body data.

```
{
  "id": "10",
  "name": "SharePoint Login",
  "description": "Credentials to login to SharePoint",
  "ownerId": "1",
  "attributes": [
    {
      "id": "58",
      "name": "Username",
      "description": "Name of the user",
      "userProvided": true,
      "masked": true,
      "createdBy": "1",
      "createdOn": "2018-07-16T06:04:32.905Z",
      "updatedBy": "1",
      "updatedOn": "2018-07-16T06:04:32.905Z",
      "version": "0"
    }
  ],
  "createdBy": "1",
  "createdOn": "2018-07-16T06:04:32.905Z",
  "updatedBy": "1",
```

```
"updatedOn": "2018-07-16T06:04:32.905Z",

"version": "0"

}
```

Parameter Description

Parameter	Description
name	Credential name in the first instance and Attribute name in the second instance
description	Credential description in the first instance and Attribute value description in second instance
userProvided	Whether the Attribute type is User-provided or Standard . If User-provided , this is set to true which is also the default parameter. For details on Attribute value types, refer Create a credential .
masked	Whether the Attribute value is Masked . This is set to true if it is encrypted. For details refer Create a credential .

2. List of credentials

This API allows you to view all credentials where a user is an owner or has access through a locker. If the user is a locker admin, it lists all credentials in the system.

1. Provide the "X-Authorization" parameter in Headers.
2. Use the POST method to access the Credential APIs with the following parameters:
http(s)://<hostname:port>/v2/credentialvault/credentials?consumed="<true or false>"&name or names="<filter credentials name(s) separated by comma>"

• **For example,**the following lists all Credentials that are **not consumed**:

```
https://crdevenv.com:81/v2/credentialvault/list?consumed=false
```

3. Click send
4. The action is successful when the response status is 200 **Found credentials**
5. You can view the response in Body data:

```
{
  "list":
  [
    { "id": 12, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn": "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "SharePoint Login", "description": "-", "lockerId": null, "ownerId": 1, "attributes": [ { "id": 61, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn": "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "Username", "description": "-", "userProvided": true, "masked": true } ], "completed": false },
    { "id": 11, "createdBy": 1, "createdOn": "2018-06-21T12:56:07+05:30", "updatedBy": 1, "updatedOn": "2018-06-21T13:00:22+05:30", "tenantId": 1, "version": 1, "name": "SharePoint-User1", "description": "name of the SharePoint user", "lockerId": null, "ownerId": 1, "attributes": [ { "id": 59, "createdBy": 1, "createdOn": "2018-06-21T12:56:07+05:30", "updatedBy": 1, "updatedOn": "2018-06-21T13:00:22+05:30", "tenantId": 1, "version": 1, "name": "SP-User1", "description": "name of the user", "userProvided": true, "masked": false }, { "id": 60, "createdBy": 1, "createdOn": "2018-06-21T13:00:22+05:30", "updatedBy": 1, "updatedOn": "2018-06-21T13:00:22+05:30", "tenantId": 1, "version": 0, "name": "Victor.Hugo", "description": null, "userProvided": true, "masked": false } ],
    "completed": false
  ]
}
```

Parameter Description

Parameter	Description
consumed	Filters credentials by fact if credential is userProvided and consumed by current user.
name	Filters credentials by a single name.
names	Filters credentials by comma separated list of names.

3. List credential by id

This API allows you to view details of the credential whose id you provide as a parameter.

Tip: You can identify the **Id** of a credential from the list of credentials that were fetched using the List Credentials API.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the GET method to access the Credential APIs with the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/credentials/<id>](http(s)://<hostname:port>/v2/credentialvault/credentials/<id>)
 - For example, the following fetches a user credential that has the id 12:
<https://crdevenv.com:81/v2/credentialvault/credentials/12>

3. Click send

4. The action is successful when the response status is 200

5. You can view the response in Body data:

```
{
  "id": 12, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
"2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "SharePoint Login", "description":
"-", "lockerId": null, "ownerId": 1,
"attributes":
[
  { "id": 61, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
"2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "Username", "description": "-",
"userProvided": true, "masked": true }
]
}
```

4. Update credential

This API allows you to update existing credentials.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use PUT method to access the Credential APIs [http\(s\)://<hostname:port>/v2/credentialvault/credentials/<id>](http(s)://<hostname:port>/v2/credentialvault/credentials/<id>)
 - For example, the following updates the credential with Id 12 :
<https://crdevenv.com:81/v2/credentialvault/credentials/12>

3. Provide the following request payload in Body:

```
{
  "id": "<Credential id>", "version": <Version id>, "name": "<Credential Name>", "description": "<Credential Description>", "lockerId":
null, "ownerId": <the user ID of the owner>,
"attributes":
[
  { "id": <Attribute id>, "version": 0, "name": "<Updated attribute name >", "description": "<Updated attribute description>", "user-
Provided": <False or true>, - Attribute type "masked": <false or true> masking is true or false }
]
}
```

- For example, if you want to update the Credential Description, you can provide following request parameters:

```
{
  "id": "12", "version": 0, "name": "SharePoint Login", "description": "Login details for
SharePoint", "lockerId": null, "ownerId": <the user ID of the owner>,
  "attributes":
  [
    { "id": 61, "version": 0, "name": "<Updated attribute name >", "description": "SharePoint User-
name", "userProvided": <False or true>, - Attribute type "masked": true
    }
  ]
}
```

Tip: You can input only those parameters that you want to update and keep the other parameter values intact; i.e. make no change.

4. Click send

5. The action is successful when the response status is 200 **Credential updated**.

6. You can view the response in Body data:

```
{
  "id": 12, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
  "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "SharePoint Login", "description":
  "Login details for SharePoint", "lockerId": null, "ownerId": 1,
  "attributes":
  [
    { "id": 61, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
    "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "Username", "description":
    "SharePoint Username", "userProvided": true, "masked": true }
  ]
}
```

Parameter Description

Parameter	Description
id	Credential id in the first instance and Attribute id in the second instance. This can be identified by using the Get Credential by Id API
version	Updated version id of Credential in the first instance and Attribute in the second instance. In case of discrepancy in earlier and current version, the conflict error is shown
name	Updated Credential in the first instance and Attribute value in the second instance
description	Updated Credential description in the first instance and Attribute value description in second instance
lockerId	Locker id if available. If no locker id is provided this is set to null which is a default value. This can be identified by using the Get Credential by Id API
ownerId	The user id of the credential owner. This can be identified by using the Get Credential by Id API
userProvided	Whether the Attribute type is User-provided or Standard . If User-provided , this is set to true which is also the default parameter. For details on Attribute value types, refer Create a credential .
masked	Whether the Attribute value is Masked . This is set to true if it is encrypted. For details refer Create a credential .

Note: Using the update credential API, neither the lockerID nor the ownerId can be changed.

5. Update ownership of credential

This API allows you to update ownership of the credential.

1. Provide the "X-Authorization" parameters in Headers.

2. Use PUT method to access the Credential APIs

http(s)://<hostname:port>/v2/credentialvault/credentials/<id>/owner/<credentialOwnerId>

• **For example**, the following changes the ownership of credential with Id **12** from owner **1** to **2**

<https://crdevenv.com:81/v2/credentialvault/credentials/12/owner/2>

3. Click send

4. The action is successful when the response status is 200 **Successful update of credential ownership**.

5. You can view the response in Body data:

```
{
  "id": 12, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
  "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "SharePoint Login", "description":
  "Login details for SharePoint", "lockerId": null, "ownerId": 1,
  "attributes":
  [
    { "id": 61, "createdBy": 1, "createdOn": "2018-06-25T14:13:58+05:30", "updatedBy": 1, "updatedOn":
    "2018-06-25T14:13:58+05:30", "tenantId": 1, "version": 0, "name": "Username", "description":
    "SharePoint Username", "userProvided": true, "masked": true }
  ]
}
```

Parameter Description

Parameter	Description
-----------	-------------

id	Id of the Credential. This can be identified by using the Get Credential by Id API
credentialOwnerId	Id of the new Credential owner. This can be identified by using the Get Credential by Id API

6. Delete credentials

This API allows you to delete the credential whose id you provide as a parameter

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to access the Credential APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/credentials/<id>`
 - **For example,** the following deleted the credential that has Id **11**
`https://crdevenv.com:81/v2/credentialvault/credentials/11`
3. Click send
4. The action is successful when the response status is **204 Successful Delete**.

APIs to manage credential attributes

As a Control Room user with **View and edit ALL credentials attributes value** use the Credential Attribute APIs to manage your Credential Attributes and its Values in the Control Room. Thus with this API, create, edit, and remove attribute values of user created credentials.

Before accessing the Credential Vault API's you must first use the authentication API and pass it as a token to use a particular Credential Vault API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

`https://crdevenv.com:81/v1/authentication`

2. Use the POST method to access the Credential APIs

For example,

`GET https://crdevenv.com:81/v2/credentialvault/credentials`

3. Provide the following request payload in Headers
 "X-Authorization" : "Authorization token"
 "Content-Type" : "application/json"

1. Create attribute value

This API allows you to create attribute value of a credential whose id you provide as a parameter

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.

2. Use the POST method to access the Credential APIs :

`http(s)://<hostname:port>/v2/credentialvault/credentials/<id>/attributevalue`

- **For example,** the following creates an attribute **username** for the credential Id **12**

`https://crdevenv.com:81/v2/credentialvault/credentials/12/username`

3. Provide the request payload in Body:

```
[
{
  "credentialAttributeId": "<Attribute Id>",
  "value": "<attribute value>"
}
```

- **For example,** for the Credential **"SharePoint Login"** that was created in the earlier example with Id **"12"**, for the Attribute **"username"** with Id **"30"** you can add the value as shown:

```
[
{
  "credentialAttributeId": "30",
  "value": "Jane.Smith"
}
```

4. Click send

5. The action is successful when the response status is 201 **Credential attribute values created**

6. You can view the response in Body data:

```
[
{
  "id":30,"createdBy":1,"createdOn":"2018-06-27T14:18:13+05:30","updatedBy":1,"updatedOn":"2018-06-27T14:18:13+05:30","tenantId":1,"version":0,"userId":null,"value":"Jane.Smith","credentialAttributeId":62
}
```

Parameter Description

Parameter	Description
credentialAttributeId	Id of the Attribute. This can be identified by using the Get Credential by Id API
value	Attribute value that you want to provide

2. List of credential attribute values

This API allows you to list all standard and user specified values for a specific credential.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to access the Credential APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/credentials/<id>/attributevalues`
 | **Forexample**, for the credential **"SharePointLogin"** with Id **"12"** for the Attribute **"username"** with Id **"30"** you can add the value as shown:
`https://crdevenv.com:81/v2/credentialvault/credentials/12/attributevalues`
3. Click send
4. The action is successful when the response status is 200 **Credential attribute values found**
5. You can view the response in Body data:


```
{ "list": [
  {
    "id": 30, "createdBy": 1, "createdOn": "2018-06-27T14:16:20+05:30", "updatedBy": 1, "updatedOn": "2018-06-27T14:16:20+05:30", "tenantId": 1, "version": 0, "userId": null, "value": "Jane.Smith", "credentialAttributeId": 62
  }
]}
```

Parameter Description

Parameter	Description
id	Id of the Credential. This can be identified by using the Get Credential by Id API
credentialAttributeId	Id of the Attribute. This can be identified by using the Get Credential by Id API
userId	Id of the Control Room user

3. Update credential attribute value

This API allows you to update the value of an attribute of the specified credential.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the PUT method to access the Credential APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/credentials/<id>/attributevalues/<attributeValueId>`
 | **For example**, for the credential **"SharePoint Login"** with Id **"12"** for Attribute **"username"** with Id **"30"** you can update the value as shown:
`https://crdevenv.com:81/v2/credentialvault/credentials/12/attributevalues/30`
3. Provide the request payload in Body:


```
[
  {
    "id": "<Attribute value id>", "version": 0, "value": "<Attribute value>", "credentialAttributeId": "<Attribute ID>"
  }
]
```
4. Click send
5. The action is successful when the response status is 200 **Credential attribute value updated**
6. You can view the response in Body data:


```
[
  {
    "id": 30, "createdBy": 1, "createdOn": "2018-06-27T14:47:55+05:30", "updatedBy": 1, "updatedOn": "2018-06-27T14:47:55+05:30", "tenantId": 1, "version": 0, "userId": null, "value": "", "credentialAttributeId": 62
  }
]
```

Parameter Description

Parameter	Description
id	Id of the Attribute Value. This can be identified by using the Get Credential by Id API
version	Version number of the updated Attribute Value
credentialAttributeld	Id of the Attribute. This can be identified by using the Get Credential by Id API

Delete credential attribute value

This API allows you to delete the attribute value of a specific credential

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to delete the creential attribute value using the Credential APIs with the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/credentials/<id>/values/<attributeValueId>](http(s)://<hostname:port>/v2/credentialvault/credentials/<id>/values/<attributeValueId>)

• **For example,**

<https://crdevenv.com:81/v2/credentialvault/credentials/12/values/30>

3. Click send
4. The action is successful when the response status is **204 Successful Delete**.

APIs to manage lockers

As a Control Room admin with **Administer all lockers permission** or a Control Room user with **Manage my lockers permission**, use the Manage Locker APIs to create, update, and delete Lockers, Consumers, and Members in the Control Room.

Before accessing these API's you must first use the authentication API and pass it as a token to use a particular Credential Vault API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

`https://crdevenv.com:81/v1/authentication`

2. Use the POST method to access the Credential APIs

For example,

`GET https://crdevenv.com:81/v2/credentialvault/lockers`

3. Provide the following request payload in Headers

`"X-Authorization" : "Authorization token"`

`"Content-Type" : "application/json"`

1. List of lockers

This API fetches list of the lockers for which the logged on user is a member (owner, manager or participant) or has usage permission. If the user is locker admin, it fetches the list of all lockers in the system.

1. Provide the "X-Authorization" parameter in Headers.
2. Use the GET method to access the Locker APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/lockers/list`

3. Click send

4. The action is successful when the response status is **200 Lockers found**

5. You can view the response in Body data:

```
{ "list":
[
{
"id": 2, "name": "SharePoint Locker", "description": "", "createdBy": 2, "createdOn": "2018-07-04T09:40:49.187Z", "updatedBy": 2, "updatedOn": "2018-07-04T09:41:04.682Z", "tenantId": 1, "version": 1
},
{
"id": 3, "name": "Outlook Locker", "description": "", "createdBy": 2, "createdOn": "2018-07-04T09:47:33.532Z", "updatedBy": 2, "updatedOn": "2018-07-04T09:47:33.532Z", "tenantId": 1, "version": 0
}
]
}
```

Parameter Description

Parameter	Description
id	Locker id
name	Name of the locker
description	Description that is given for the locker
consumed	Whether consumed by logged in user. This could be either True or False

2. Create a locker

This API allows you to create a locker.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to access the Locker APIs :
`http(s)://<hostname:port>/v2/credentialvault/lockers`

3. Provide the request payload in Body:

```
{
  "name": "<Locker name>",
  "description": "<Locker description>"
}
```

For example, the following creates a "SalesForce" locker:

```
{
  "name": "SalesForce Locker", "description": "Use this to store all credentials for the
  Salesforce application"
}
```

4. Click send

5. The action is successful when the response status is 201 **Locker created**

6. You can view the response in Body data:

```
{
  "id": 4, "name": "SalesForce Locker", "description": "Use this to store all credentials for the
  Salesforce application", "createdBy": 2, "createdOn": "2018-07-04T09:55:04.045Z", "updatedBy": 2,
  "updatedOn": "2018-07-04T09:55:04.045Z", "tenantId": 1, "version": 0
}
```

Parameter Description

Parameter	Description
name	Name of the locker as provided by the Locker admin
description	Description of the locker

3. List of all lockers with additional fields included

This API fetches list of the lockers for which the logged on user is a member (owner, manager or participant) or has usage permission. If the user is locker admin, it fetches the list of all lockers in the system. The response includes locker members and count of credentials inside each locker.

1. Provide the "X-Authorization" parameter in Headers.

2. Use the GET method to access the Locker APIs with the following parameters:

[http\(s\)://<hostname:port>/v2/credentialvault/lockers/list](http(s)://<hostname:port>/v2/credentialvault/lockers/list)

For example, the following lists all locker names with additional fields

<https://crdevenv.com:81/v2/credentialvault/lockers/list>

3. Click send

4. The action is successful when the response status is 200 **Lockers found**

5. You can view the response in Body data:

```
{ "list": [
  { "id": 2, "name": "SharePoint Locker", "description": "", "createdBy": 2, "createdOn": "2018-07-
  04T09:40:49.187Z", "updatedBy": 2, "updatedOn": "2018-07-04T10:10:04.163Z", "tenantId": 1, "version":
  2, "members": [ { "id": 1, "permissions": [ "participate", "manage" ] }, { "id": 2, "permissions": [
  "participate", "own", "manage" ] } ], "countOfCredentials": 1 },

  { "id": 3, "name": "Outlook Locker", "description": "", "createdBy": 2, "createdOn": "2018-07-
  04T09:47:33.532Z", "updatedBy": 2, "updatedOn": "2018-07-04T09:47:33.532Z", "tenantId": 1, "version":
  0, "members": [ { "id": 2, "permissions": [ "participate", "own", "manage" ] } ], "coun-
  tOfCredentials": 0 },

  { "id": 4, "name": "SalesForce Locker", "description": "-", "createdBy": 2, "createdOn": "2018-07-
```

```
04T09:55:04.045Z", "updatedBy": 2, "updatedOn": "2018-07-04T09:55:04.045Z", "tenantId": 1, "version":
0, "members": [ { "id": 2, "permissions": [ "participate", "own", "manage" ] } ], "countOfCredentials": 0 }
}]}
```

4. List locker by id

This API allows you to fetch locker details based on its Id.

Tip: You can identify the **Id** of a locker from the list of lockers that were fetched using the List Lockers API.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the GET method to access the Credential APIs with the following parameters:
http(s)://<hostname:port>/v2/credentialvault/lockers/<id>

• **For example**, the following fetches a locker that has the id **2**:

```
https://crdevenv.com:81/v2/credentialvault/lockers/2
```

3. Click send
4. The action is successful when the response status is **200 Locker**.
5. You can view the response in Body data:

```
{
  "id": 2, "name": "SharePoint Locker", "description": "", "createdBy": 2, "createdOn": "2018-07-04T09:40:49.187Z", "updatedBy": 2, "updatedOn": "2018-07-04T10:10:04.163Z", "tenantId": 1, "version": 2
}
```

5. Update locker

This API allows you to update the values of a specific locker

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the PUT method to access the Credential APIs with the following parameters:
http(s)://<hostname:port>/v2/credentialvault/lockers/<id>

• **For example**, for the locker **"SharePoint Locker"** with Id**"2"** you can update the value as shown:

```
https://crdevenv.com:81/v2/credentialvault/lockers/2
```

3. Provide the request payload in Body:

```
{
  "id": "<locker id>", "name": "<updated lockername>", "description": "<updated description>", "version": 0
}
```
4. Click send
5. The action is successful when the response status is **200 Locker updated**
6. You can view the response in Body data:

```
{
  "id": 2, "createdBy": 8, "createdOn": "2018-06-16T10:59:33Z", "updatedBy": 8, "updatedOn": "2018-06-16T10:59:37Z", "version": 1, "name": "SharePoint-Locker", "description": "Locker to manage SharePoint credentials"
}
```

Parameter Description

Parameter	Description
id	Id of the locker. This can be identified by using the Get Locker by Id API
version	Version number of the updated locker value
name	Updated name of the locker
description	Updated description of the locker

6. Delete locker

This API allows you to delete the locker whose id you provide as a parameter

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to remove the credentials of a locker using Locker APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/lockers/<id>`
 - **For example**, the following deletes the credential that has Id **4**
`https://crdevenv.com:81/v2/credentialvault/lockers/4`
3. Click send
4. The action is successful when the response status is **204 Successful Delete**.

7. List all locker consumers

This API allows you to fetch the list of roles that can use credentials in the locker

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to access the Locker APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/lockers/<id>/consumers`
 - **For example**, the following fetches the roles that can use credentials from the locker with Id **2**
`https://crdevenv.com:81/v2/credentialvault/lockers/2/consumers`
3. Click send
4. The action is successful when the response status is **200 Successful**
5. You can view the response in the Body data:

```
{
  "list":
  [
    { "id": "14" }
  ]
}
```

8. Add locker consumers

This API allows you to add a consumer to the specified locker

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to access the Locker APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/lockers/<id>/consumers`
 - **Forexample**, the following allows you to connect to the locker with Id **2** wherein you want to add the role with consumer id **14**
`https://crdevenv.com:81/v2/credentialvault/lockers/2`
3. Provide the request payload in Body:


```
{
  "id": "<consumer id>"
}
```

 - **For example**,


```
{
    "id": 14
  }
```
4. Click send
5. The action is successful when the response status is **204 Successful creation of locker consumer**

Parameter Description

Parameter	Description
id	Id of the locker consumer. This can be identified by using the Get Locker by Id API

9. Remove locker consumers

This API allows you to remove consumers from the specified locker

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to remove a locker consumer using Locker APIs with the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/consumers/<roleId>](http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/consumers/<roleId>)

▪ **For example**, the following deletes the credential that has Id 2

<https://crdevenv.com:81/v2/credentialvault/lockers/2/consumers/14>

3. Click send
4. The action is successful when the response status is **204 Successful Delete**.

10. List all locker members

This API allows you to fetch the list of all users that can add/remove credentials from the specified locker or edit the locker settings.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to access the Locker APIs with the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/members](http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/members)

▪ **For example**, the following fetches the users that can add/remove user from the locker with Id 2

<https://crdevenv.com:81/v2/credentialvault/lockers/2/members>

3. Click send
4. The action is successful when the response status is **200 Locker Members**
5. You can view the response in the Body data:

```
{
  "list":
  [
    {
      "id": 1, "permissions": [ "participate", "manage" ]
    },
    {
      "id": 2, "permissions": [ "participate", "own", "manage" ]
    }
  ]
}
```

11. Add or update locker member

This API allows you to add a user who can add/remove credentials from the locker or edit the locker settings.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the PUT method to add/update user using Locker APIs with the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/members/<userId>](http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/members/<userId>)

▪ **Forexample**, the following allows you to connect to the locker with Id 2 wherein you want to add the user with member id 3

<https://crdevenv.com:81/v2/credentialvault/lockers/2/members/3>

3. Provide the request payload in Body:

```
{
  "permissions":
  ["<permission type>"]
}
```

▪ **For example**,

```
{
  "permissions":
  ["participate", "manage"]
}
```

4. Click send
5. The action is successful when the response status is **204 Successful creation/update of locker member**

Parameter Description

Parameter	Description
permissions	Permission that the user is granted. It can be participate , manager and/or own

12. Remove locker member

This API allows you to remove locker users or members from the specified locker

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to remove a locker consumer using Locker APIs with the following parameters:
http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/members/<userId>

• **For example**, the following deletes the locker user that has Id 3

`https://crdevenv.com:81/v2/credentialvault/lockers/2/members/3`

3. Click send
4. The action is successful when the response status is 204 **Successful removal of locker member**

13. Add credential to a locker

This API allows you to add your own credential to a locker where you have owner, manage or participant permissions.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the PUT method to add credentials to a locker using the Credentials API with following parameters:
http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/credentials/<credentialId>

• **For example**, the following adds a credential to the locker with Id 2

`https://crdevenv.com:81/v2/credentialvault/lockers/2/credentials/58`

3. Click send
4. The action is successful when the response status is 200 **Credential has been added to the locker.**

14. List all locker credentials

This API allows you to fetch the list of credetials in a locker for which you have owner, manage or participant permissions.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch the list of credentials using the Credentials API with following parametes:
http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/credentials

• **For example**, the following lists all credentials that are added to the locker with Id 2

`https://crdevenv.com:81/v2/credentialvault/lockers/2/credentials`

3. Click send
4. The action is successful when the response status is 200 **Locker credentials**
5. You can view the response in the Body data:


```
{ "list": [
  {
    "id": 10, "name": "SharePoint Credentials", "description": "", "completed": false, "lockerId": 2,
    "ownerId": 2, "attributes":
    [
      {
        "id": 58, "name": "username", "description": "", "userProvided": true, "masked": false, "createdBy":
        2, "createdOn": "2018-07-04T10:09:32.854Z", "updatedBy": 2, "updatedOn": "2018-07-04T10:09:32.854Z",
        "tenantId": 1, "version": 0
      },
      {
        "id": 59, "name": "password", "description": "", "userProvided": true, "masked": true, "createdBy":
        2, "createdOn": "2018-07-04T10:09:32.854Z", "updatedBy": 2, "updatedOn": "2018-07-04T10:09:32.854Z",
        "tenantId": 1, "version": 0
      }
    ]
  },
  {
    "createdBy": 2, "createdOn": "2018-07-04T10:09:32.854Z", "updatedBy": 2, "updatedOn": "2018-07-
    04T10:10:04.316Z", "tenantId": 1, "version": 1
  }
]
```



```
}  
] }
```

15. Remove credential from a locker

This API allows you to remove locker users or members from the specified locker

1. Provide the "X-Authorization" parameters in Headers.
2. Use the DELETE method to remove a locker consumer using Locker APIs with the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/lockers/<lockerId>/credentials/<credentialId>`
 - **For example**, the following deletes the credential with Id **59** from the locker with Id **2**
`https://crdevenv.com:81/v2/credentialvault/lockers/2/credentials/59`
3. Click send
4. The action is successful when the response status is 204 **redential has been removed from the locker**

APIs to manage credential vault mode

As a Control Room admin with **View and manage settings** permission, use the Manage Credential Vault mode APIs to manage the CV mode of the Control Room.

Before accessing these API's you must first use the authentication API and pass it as a token to use a particular Credential Vault API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

`https://crdevenv.com:81/v1/authentication`

2. Use the POST method to access the Credential APIs

For example,

`GET https://crdevenv.com:81/v2/credentialvault/mode`

3. Provide the following request payload in Headers

`"X-Authorization" : "Authorization token"`

`"Content-Type" : "application/json"`

View current credential vault mode

This API allows you to view the current mode used to connect to the Credential Vault. This can either be "Express" or "Manual"

1. Provide the "X-Authorization" parameters in Headers.
2. Use GET method to access the Credential Vault API's using the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/mode`

▮ **For example,** the following displays the current mode used to connect to the Credential Vault:

`https://crdevenv.com:81/v2/credentialvault/mode`

3. Click Send/Execute
4. The action is successful when the response status is **200 CV Mode has been successfully received**
5. You can view the response in the Body data:

```
{
  "name": "Express"
}
```

Update Credential Vault mode

This API allows you to change the mode used to connect to the Credential Vault. This can either be "Express" or "Manual"

1. Provide the "X-Authorization" parameters in Headers.
2. Use PUT method to access the Credential Vault API's using the following parameters:
`http(s)://<hostname:port>/v2/credentialvault/mode`

▮ **For example,** the following displays the current mode used to connect to the Credential Vault:

`https://crdevenv.com:81/v2/credentialvault/mode`

3. Provide the request payload in the Body:

```
{
  "name": "<mode>",
  "privateKey": "<CV Key>"
}
```

▮ **For example,**

```
{
  "name": "Manual",
  "privateKey": "ABC123"
}
```

4. Click Send/Execute
5. The action is successful when the response status is **204 Mode has been successfully set**

Parameter Description

Parameter	Description
name	Connection type - Manual or Express. This is specified during the initial configuration of the Control Room. Refer Control Room Settings for updating this from the Control Room UI.
privateKey	The Master key that is used to connect to credential vault.

Check if the master key has been applied or not

This API allows you to verify whether the Control Room Master key that connects to Credential Vault has been applied or not. Use this to check the Credential Vault status.

1. Provide the "X-Authorization" parameters in Headers.
2. Use GET method to access the Credential Vault API's using the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/keys/private](http(s)://<hostname:port>/v2/credentialvault/keys/private)

For example,

<https://crdevenv.com:81/v2/credentialvault/keys/private>

3. Click Send/Execute
4. The action is successful when the response status is **200 Private key status**
5. You can view the response in the Body data:

```
{
  "applied": true
}
```

Note: The value is shown **false** if the private key was not applied.

Apply Master key to unlock Credential Vault after restarting Control Room in Manual mode

This API allows you to unlock the Credential Vault after restarting the Control Room using the Manual mode of connection. Use this to verify whether the Credential Vault is connected.

1. Provide the "X-Authorization" parameters in Headers.
2. Use PUT method to access the Credential Vault API's using the following parameters:
[http\(s\)://<hostname:port>/v2/credentialvault/keys/private](http(s)://<hostname:port>/v2/credentialvault/keys/private)

For example,

<https://crdevenv.com:81/v2/credentialvault/keys/private>

3. Provide the request payload in the Body:

```
{
  "privateKey": "<CV Key>"
}
```
4. Click Send/Execute
5. The action is successful when the response status is **200 Private key has been successfully applied**

Parameter Description

Parameter	Description
privateKey	The Master key that is used to connect to credential vault. This is specified during the initial configuration of the Control Room. You can also change the mode from Control Room Settings by providing this key.

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200/204	Action completed successfully	NA
304	No changes applied	Update as required
400	Bad request parameter	Retry with valid parameters
401	Authentication required	Provide authentication parameters. For example, X-Authorization key
403	Unauthorized access	Ensure you are authorized to access the Control Room
404	Not found	Ensure that the request payload is available in Control Room
409	Conflict	Ensure the parameters provided are correct
500	Internal server error	Ensure the server is up and running
501	Permission error	Ensure that you have the required permission

Audit Logs

All API activities are logged in the Control Room Audit Log page. As a Control Room administrator or a user with View everyone's audit log actions, you can view the audit entries as shown:

Audit log

Audit log

Time filter: Last 7 days

Action type

Choose action type

Action type: Add lockerAction type: Add credentialAction type: Update CredentialAction type: Credential Vault SettingsAction type: Configure Credential VaultAction type: Connect Credential Vault

Actions (9 of 113)

	STATUS	TIME	ACTION TYPE	ITEM NAME	ACTION TAKEN BY	SOURCE	REQUEST ID	SOURCE DEVICE	
<input type="checkbox"/>	Unsuccessful	16:53:22 IST 2018-07-05	Connect Credential Vault	Express	N/A	Control Room	516b40f1-e059-45a8-ac3...		⋮
<input type="checkbox"/>	Unsuccessful	16:50:28 IST 2018-07-05	Connect Credential Vault	Express	N/A	Control Room	4180cf93-3c12-4f10-89e2...		⋮
<input type="checkbox"/>	Successful	16:50:28 IST 2018-07-04	Add credential	Outlook Credentials	mike.lee	Control Room	270c9be7-b0e7-429c-a2b...		⋮
<input type="checkbox"/>	Successful	15:39:32 IST 2018-07-04	Add credential	SharePoint Credentials	mike.lee	Control Room	8a0824de-f8b8-4e18-8b9...		⋮
<input type="checkbox"/>	Successful	15:25:04 IST 2018-07-04	Add locker	SalesForce Locker	mike.lee	Control Room	963ce57d-a9aa-41be-aa21...		⋮
<input type="checkbox"/>	Successful	15:17:33 IST 2018-07-04	Add locker	Outlook Locker	mike.lee	Control Room	9572a3b2-11f3-4f0b-aba5...		⋮
<input type="checkbox"/>	Successful	15:10:49 IST 2018-07-04	Add locker	SharePoint	mike.lee	Control Room	3ff440dd-b8fb-4c46-a33c...		⋮
<input type="checkbox"/>	Successful	17:52:34 IST 2018-07-02	Connect Credential Vault	Express	System	Control Room	3b5de874-d3bc-4272-a4b...		⋮
<input type="checkbox"/>	Successful	16:59:26 IST 2018-07-02	Connect Credential Vault	Express	System	Control Room	3e6d8842-bf5e-4dad-aa5...	automationanywhere	⋮

The following illustration is that of a sample audit log details page for creating a locker with the Locker API:

Audit log > View action

Add Locker

< Back

ACTION DETAILS

Status
Successful

Action taken by
mike.lee

Object type
Action

Source device

Request ID
3ff440dd-b8fb-4c46-a33c-215754eda3c1

Item name
SharePoint

Time
15:10:49 IST
2018-07-04

Action type
Add Locker

Source
Control Room

ADD LOCKER DETAILS

ATTRIBUTE	VALUE
Locker Name	SharePoint
Description	--
Locker Owners	mike.lee
Locker Managers	--
Locker Participants	--
Locker Consumers	--
No. of Credentials	0

11. API for data migration from 10.x Control Room to 11.x Control Room

As a Control Room administrator with **View and Manage Migration** role permissions, use the Migration APIs to migrate data from 10.x Control Room to the current Control Room version 11.2.

The Migration APIs allow you to,

1. Save / update and connect to the 10.x Control Room database
2. Save / update and connect to the 2.x Bot Insight database, if available
3. Specify option to migrate data based on Roles, Users or Bots
4. Fetch list of data based on option specified for migration i.e. Roles, Users, or Bots
5. View the migration progress summary
6. View migration statistics of number of entities that succeeded / failed per migration
7. Fetch list of new and updated bots from 10.x Control Room post migration
8. Migrate files in bulk from the 10.x Control Room My docs folder post migration

Alternately, you can use the Migration wizard given in Administration > Migration module to migrate the data from the Control Room user interface. Refer [Migration Overview](#) for details.

Note: The examples provided in this article are for reference only.

API End Point

Use the following end points to access the API:

1. For [migration process](#) use **<Control Room URL>/v2/migration**
2. For migrating files from the **My Docs** folder of source 10.x Control Room [after the migration process has completed](#) use **<Control Room URL>/v1/migration**

For example,

`https://crdevenv.com:81/v2/migration`

Migration Process APIs

The Migration APIs allow you to migrate 10.x Control Room data to 11.x Control Room using the end point mentioned earlier.

Before accessing the Migration API's you must first use the authentication API and pass it as a token to use a particular Migration API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

`https://crdevenv.com:81/v1/authentication`

2. Provide the following request payload in Headers
 "X-Authorization" : "Authorization token"
 "Content-Type" : "application/json"
3. Provide the following request payload in Body:

```
{
  "username": "<Username>",
  "password": "<Password>"
}
```

For example,

```
{
  "username": "Ellie.Brown",
  "password": "12345678"
}
```

1. Connect to source Control Room database

This API allows you to save and update the connection configuration to the source 10.x Control Room database.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the POST method to connect to the 10.x Control Room database using the end point **http(s)://<host-name>:port/v2/migration/connection**
 - For example,
`https://crdevenv.com:81/v2/migration/connection`
4. Provide the following request parameters in Body:


```
{
  "host": "string", "port": 0, "databaseName": "string", "username": "string", "password": "string", "integratedSecurity": true, "encrypt":
  true, "privateKey": "string", "repoPath": "string"
}
```

 - For example,


```
{
    "host": "PRODUCTLT",
    "port": 1433,
    "databaseName": "CR104MIG",
    "username": "Ellie.Brown",
    "password": "12345678",
    "integratedSecurity": true,
    "encrypt": true,
    "privateKey": "ABC123",
    "repoPath": "D:\\Data\\Automation Anywhere Server Files"
  }
```
5. Click **Send**
6. The connection parameters are successfully saved when the response status is 200 **Successful operation**.

Parameter Description

Parameter	Description
host	Source Control Room database host name
port	Source Control Room database port number
databaseName	Source Control Room database name
username	Username to connect to database
password	Password to connect to database
integratedSecurity	An indicator whether to use Windows authentication when connecting to source database. Set this to true if you want use Windows authentication. The default value is false
encrypt	An indicator whether to use secure connection to source database. Set this to true if you want to use a secure connection. The default value is false
privateKey	The private key to decrypt credential values in source database. This is available for configuration during the initial Control Room setup.
repoPath	The shared repository path where Control Room 10.x repository is stored

2. Get stored connection details

This API allows you to fetch the stored connection details of source 10.x Control Room database from where the data can be migrated.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the GET method to fetch the connection configuration of 10.x Control Room database using the end point **http(s)://<host-name>:port/v2/migration/connection**
 - For example,
`https://crdevenv.com:81/v2/migration/connection`

4. Click **Send**

5. The connection parameters are successfully saved when the response status is 200 **Migration config.**

6. You can view the result in Body data:

```
{
  "host": "productlt",
  "port": 1433,
  "databaseName": "CR104MIG",
  "username": "",
  "password": "",
  "integratedSecurity": true,
  "encrypt": false,
  "privateKey": "",
  "repoPath": "D:\\\\DATA\\AUTOMATION ANYWHERE SERVER FILES"
}
```

Parameter Description

Parameter	Description
host	Source database host
port	Source database port
databaseName	Source database name
username	Username to connect to source database
password	Password to connect to source database
integratedSecurity	An indicator whether to use Windows authentication when connecting to source database, default value is false
encrypt	An indicator whether to use secure connection to source database, default value is false
privateKey	Private key to decrypt credential values in source database
repoPath	The shared repository path where Control Room 10.x repository is stored

3. Connect to 2.x Bot Insight database, if available

This API allows you to connect to the source 2.x Bot Insight database if available to migrate analytics data.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the POST method to connect to the BotInsight database using the end point **http(s)://<host-name:port>/v2/migration/connection /botinsight**

For example,

`https://crdevenv.com:81/v2/migration/connection/botinsight`

4. Provide following request parameters in Body:

```
{
  "host": "string",
  "port": 0,
  "databaseName": "string",
  "username": "string",
  "password": "string",
  "integratedSecurity": true,
  "encrypt": true,
  "serverUrl": "string"
}
```

For example,

```
{
  "host": "Productlt",
  "port": 8091,
  "databaseName": "BotInsight",
  "username": "Ellie.Brown",
  "password": "12345678",
}
```



```
"integratedSecurity": true,
"encrypt": true,
"serverUrl": "https://productlt.aae.com:82/analytics"
}
```

5. Click **Send**

6. The connection parameters are successfully saved when the response status is 200 **Successful operation**.

Parameter Description

Parameter	Description
host	Source Bot Insight database host name
port	Source Bot Insight database port number
databaseName	Source Bot Insight database name
username	Username to connect to database
password	Password to connect to database
integratedSecurity	An indicator whether to use Windows authentication when connecting to source database. Set this to true if you want use Windows authentication. The default value is false
encrypt	An indicator whether to use secure connection to source database. Set this to true if you want to use a secure connection. The default value is false
serverUrl	Server url where the Bot Insight Visualization ServerPort

4. Get stored connection details

This API allows you to fetch the stored connection details of source 2.x Bot Insight database from where the data can be migrated.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the GET method to fetch the connection configuration of 10.x Control Room database using the end point **http(s)://<host-name:port>/v2/migration/connection/botinsight**

• For example,

```
https://crdevenv.com:81/v2/migration/connection/botinsight
```

4. Click **Send**

5. The connection parameters are successfully saved when the response status is 200 **Migration config**.

6. You can view the result in Body data:

```
{
"host": "Productlt",
"port": 8091,
"databaseName": "BotInsight",
"username": "Ellie.Brown",
"password": "12345678",
"integratedSecurity": true,
"encrypt": true,
"serverUrl": "https://productlt.aae.com:82/analytics"
}
```

Parameter Description

Parameter	Description
host	Source Bot Insight database host name
port	Source Bot Insight database port number
databaseName	Source Bot Insight database name
username	Username to connect to database
password	Password to connect to database
integratedSecurity	An indicator whether to use Windows authentication when connecting to source database. Set this to true if you want use Windows authentication. The default value is false

encrypt	An indicator whether to use secure connection to source database. Set this to true if you want to use a secure connection. The default value is false
serverUrl	Server url where the Bot Insight Visualization ServerPort

5. List of entities of TYPE available for migration in source database

This API returns list of entities available for migration in source database by TYPE parameter. Using the either of the options - Role or User, you can migrate **all** data that is associated with the parameter chosen.

Note: For selective migration of data i.e. selecting only certain data based on specified parameter, use the [Migration wizard](#) provided in the Control Room.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to connect to the Control Room database using the end point **http(s)://<host-name:port>/v2/migration/connection /entities** followed by TYPE parameter that could include any one of the option - **Roles, Users, Bots, or Schedules**

For example,

<https://crdevenv.com:81/v2/migration/connection/entities?Type=ROLE>

3. Click Send
4. The data is migrated successfully when the response status is 200 **Collection of items from source db eligible for migration**
5. The list of entities based on TYPE parameter are displayed in Body.

```
{
  "entities":
  [
    { "id": "0", "type": "ROLE", "sourceId": "1", "targetId": "0", "name": "Admin", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "2", "targetId": "0", "name": "Basic", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "3", "targetId": "0", "name": "IQBotValidator", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "4", "targetId": "0", "name": "AnalyticsExperts", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "5", "targetId": "0", "name": "AnalyticsConsumers", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "6", "targetId": "0", "name": "BotAgentUser", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "7", "targetId": "0", "name": "BotFarmAdmin", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "8", "targetId": "0", "name": "IQBotServices", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "9", "targetId": "0", "name": "Bot Creator 10x", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "10", "targetId": "0", "name": "Bot Runner 10x", "status": "SUCCESS", "reason": "" },
    { "id": "0", "type": "ROLE", "sourceId": "11", "targetId": "0", "name": "Bot Scheduler 10x", "status": "SUCCESS", "reason": "" }
  ]
}
```

Parameter Description

Parameter	Description
id	Migration ID
type	Type of entity selected for migration - Role, User or Bot
sourceId	Id of entity in the source database
targetId	Id of entity after Migration in the target database
name	Name of the entity in the source database
status	The migration status for that particular entity

reason	The reason for migration failure for that particular entity
--------	---

6. Prepare migration data based on User input

This API allows you to migrate entities with associated data based on the sub-section of the entity type specified for migration.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the POST method to migrate the data using the endpoint **http(s)://<hostname:port>/v2/migration/prepare**

For example,

`https://crdevenv.com:81/v2/migration/prepare`

4. Provide following request payload in Body:

```
{
  "selected":
  [
    { "type": "<entity type>",
      "sourceId": "string" }
  ],
  "excludes": [ "<entity type>"
  ]
  "excludeMetaBots":true,
  "overwriteFiles":true
}
```

For example,

```
{ "selected": [ { "type": "ROLE", "sourceId": "12" } ], "excludes": [ "BOT" ] }
```

5. Click Send
6. The data is listed successfully for migration when the response status is 200
7. The result is displayed in the Body

```
{ "selected":
[
{ "type": "ROLE", "sourceId": "12" } ],
"excludes": [ "BOT" ]
}
```

Parameter Description

Parameter	Description
type	Type of entity selected for migration - Role, User or Bot and Schedules
sourceId	The id of the entity in the source database
excludes	The entity name that is excluded from migration. The options are available based on the entity type selected. Hence, when you select Role or User you can Exclude Bots and Schedules ; when you select Bots and Schedules , you can Exclude MetaBots , and/or Overwrite existing Bots .
excludeMetaBots	The flag to choose whether to migrate MetaBots.
overWriteFiles	The flag to choose whether to overwrite the File (if the same File is present at the destination)

7. List the data for migration

This API allows you to fetch the list of entities that are ready for migration.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the GET method to fetch the list of entities using the endpoint **http(s)://<hostname:port>/v2/migration/prepped**

For example,

`https://crdevenv.com:81/v2/migration/prepped`

4. Click Send
5. The data is listed successfully for migration when the response status is 200

6. The result is displayed in the Body:

```
{
  {
    "ROLE":
    { "entities":
    [
      { "id": "0", "type": "ROLE", "sourceId": "11", "targetId": "0", "name": "Bot Scheduler 10x",
        "status": "SUCCESS", "reason": "" }
    ] },
    "USER":
    { "entities": [ { "id": "0", "type": "USER", "sourceId": "2", "targetId": "0", "name":
      "Tom.Hardy", "status": "SUCCESS", "reason": "" }, { "id": "0", "type": "USER", "sourceId": "12",
        "targetId": "0", "name": "P.G.Wodehouse", "status": "SUCCESS", "reason": "" }
    ] },
    "CREDENTIAL": { "entities": [] },
    "BOT": { "entities": [] },
    "SCHEDULE": { "entities": [] }
  }
}
```

Parameter Description

Parameter	Description
id	Migration ID
type	Type of entity selected for migration - Role, User or Bot
sourceId	Id of entity in the source database
targetId	Id of entity after Migration in the target database
name	Name of the entity in the source database
status	The migration status for that particular entity
reason	The reason for migration failure for that particular entity

8. Start Migration

This API allows you to launch the migration process.

1. Provide the "X-Authorization" parameters in Headers.
2. Provide credential parameters in Body
3. Use the POST method to migrate the data using the endpoint **http(s)://<hostname:port>/v2/migration/start/async**

• For example,

https://crdevenv.com:81/v2/migration/start/async

4. Click Send
5. The data migration starts successfully when the response status is 200 **Successful operation**
6. The result is displayed in the Body data

```
{
  "id": 1,
  "name": "2018.07.17.16.13.48.ellie.brown",
  "createdBy": 1,
  "migrationType": "ROLE_EXCLUDE_BOT_SCHEDULE"
}
```

Parameter Description

Parameter	Description
id	Migration ID
name	Name of the user who initiated the migration
createdBy	Id of the entity that started the migration i.e. the Control Room administrator
migrationType	The migration type chosen - Role, User, or Bots and Schedules

10. Migration object by id

This API lists the migration object details based on the ID that is generated using the Start Migration API.

1. Provide the "X-Authorization" parameters in Headers.
2. Provide credential parameters in Body
3. Use the GET method to fetch object details by id using the endpoint **http(s)://<hostname:port>/v2/migration/<id>**

• **For example,**

`https://crdevenv.com:81/v2/migration/9`

4. Click Send
5. The object details are listed successfully when the response status is **200**
6. The details are shown in the Body data:

```
{
  "id": "9",
  "name": "2018.07.17.16.13.48.ellie.brown",
  "startTime": "2018-07-17T10:43:48.127Z",
  "endTime": "2018-07-17T10:43:49.833Z",
  "createdBy": "1",
  "migrationType": "ROLE_EXCLUDE_BOT_SCHEDULE",
  "entities": []
}
```

Parameter Description

Parameter	Description
id	Migration ID
name	Name of the user who initiated the migration
startTime	The time when the migration was initiated
endTime	The time when the migration was complete
createdBy	Id of the entity that started the migration i.e. the Control Room administrator
entities	List of entities migrated during this Migration process
migrationType	The migration type chosen - Role, User, or Bots and Schedules

11. Migration Progress

This API allows you to view the process of migration that is in progress.

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Provide credential parameters in Body
3. Use the GET method to fetch object details by id using the endpoint **http(s)://<hostname:port>/v2/migration/pogress**

• **For example,**

`https://crdevenv.com:81/v2/migration/progress`

4. Click Send
5. The object details are listed successfully when the response status is **200**
6. The details are shown in the Body data:

```
{
  "migration":
```

```
{ "id": "10", "name": "2018.07.17.16.55.59.ellie.brown", "startTime": "2018-07-17T11:25:59.800Z",
"endTime": "2018-07-17T11:26:16.002Z", "createdBy": "1", "migrationType": "BOT_EXCLUDE_METABOT",
"entities": [] },
"current": "SCHEDULE",
"progress":
{
"BOT": { "total": "10", "successful": "7", "failed": "0", "skipped": "3" },
"SCHEDULE": { "total": "8", "successful": "8", "failed": "0", "skipped": "0" }
}
}
```

Parameter Description

Parameter	Description
id	Migration id
name	Migration name displayed
startTime	Timestamp when Migration process started
endTime	Timestamp when Migration process completed. Null when migration is in progress
createdBy	Id of the user who created/started the migration process
current	Type of entity currently being migrated - ROLE, USER, CREDENTIAL, BOT, or SCHEDULE]
progress	Progress of the entities - <ul style="list-style-type: none"> total - total number of entities of specific type to be migrated successful - number of entities out of total migrated successfully failed - number of entities out of total failed to be migrated skipped - number of entities out of total skipped during migration

12. Migration statistics - number of entities that succeeded / failed per Migration This

API allows you to view the number of successful or failed entities per migration.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch object details by id using the endpoint **http(s)://<hostname:port>/v2/migration/statistics**

• **For example,**

<https://crdevenv.com:81/v2/migration/statistics>

3. Click Send
4. The object details are listed successfully when the response status is **200**
5. The details are shown in the Body data:

```
{
"items": [
{ "id": "1", "name": "2018.07.13.11.14.59.ellie.brown", "startTime": "2018-07-13T05:44:59.787Z",
"endTime": "2018-07-13T06:56:25.537Z", "createdBy": "1", "duration": "4285s", "numSuccess": 0,
"numFailed": 0, "numSkipped": 0 },
{ "id": "2", "name": "2018.07.13.12.28.08.ellie.brown", "startTime": "2018-07-13T06:58:09.283Z",
"endTime": "2018-07-13T06:58:12.910Z", "createdBy": "1", "duration": "3s", "numSuccess": 1,
"numFailed": 1, "numSkipped": 0 },
{ "id": "3", "name": "2018.07.13.12.40.34.ellie.brown", "startTime": "2018-07-13T07:10:34.470Z",
"endTime": "2018-07-13T07:10:40.060Z", "createdBy": "1", "duration": "5s", "numSuccess": 10,
"numFailed": 0, "numSkipped": 0 },
{ "id": "4", "name": "2018.07.13.12.42.19.ellie.brown", "startTime": "2018-07-13T07:12:20.007Z",
"endTime": "2018-07-13T07:12:23.107Z", "createdBy": "1", "duration": "3s", "numSuccess": 0,
"numFailed": 0, "numSkipped": 6 },
{ "id": "5", "name": "2018.07.13.13.39.53.ellie.brown", "startTime": "2018-07-13T08:09:53.113Z",
"endTime": "2018-07-13T08:10:02.673Z", "createdBy": "1", "duration": "9s", "numSuccess": 4,
"numFailed": 0, "numSkipped": 0 }
]
}
```

Parameter Description

Parameter	Description
id	migration id
name	Migration name displayed
startTime	Timestamp when Migration process started
endTime	Timestamp when Migration process completed. Null when migration is in progress
createdBy	Id of the user who created the object
duration	Duration of migration - seconds or nano seconds
numSuccess	Number of items successfully migrated
numFailed	Number of items that failed to migrate
numSkipped	Number of items that were skipped during migration

Post Migration process APIs

Use the Migration APIs after the process has completed to

1. Import files from the My Docs folder of 10.x Control Room
2. Fetch the list of new or modified bots from 10.x Control Room since last migration run

Before accessing the APIs you must first use the authentication API and pass it as a token to use a particular Migration API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

For example,

https://crdevenv.com:81/v1/authentication

2. Provide the following request payload in Headers
 "X-Authorization" : "Authorization token"
 "Content-Type" : "application/json"
3. Provide the following request payload in Body:

```
{
  "username": "<Username>",
  "password": "<Password>"
}
```

For example,

```
{
  "username": "Ellie.Brown",
  "password": "12345678"
}
```

Important: When the error code 404 is shown while using any or all the post migration APIs, re-use the API to fetch the list of root folders from 10.x Control Room i.e. **http(s)://<host-name:port>/v1/migration/legacyrepository/rootDirectories**.

A. Migrate files from My Docs folder

Use certain set of APIs to migrate files from the My Docs folder of the 10.x Control Room. These APIs allow you to migrate large number of files that could either be used in bots as dependencies or though stand alone are useful for automation.

1. Fetch list of root folders from 10.x Control Room

This API allows you to fetch the list of folders available in the 10.x Control Room Repository. This will help you understand the folder structure that was available in the source Control Room.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch object details by id using the endpoint **http(s)://<host-name:port>/v1/migration/legacyrepository/rootDirectories** followed by **excludeMetaBot** parameter

For example,

https://crdevenv.com:81/v1/migration/legacyrepository/rootDirectories?excludeMetaBot=true

3. Click Send
4. The object details are listed successfully when the response status is **200**
5. The details are shown in the Body data:


```
[
  { "name": "My Docs", "path": "Automation Anywhere\\My Docs" },
  { "name": "My Exes", "path": "Automation Anywhere\\My Exes" },
  { "name": "My Reports", "path": "Automation Anywhere\\My Reports" },
  { "name": "My Scripts", "path": "Automation Anywhere\\My Scripts" },
  { "name": "My Tasks", "path": "Automation Anywhere\\My Tasks" },
  { "name": "My Workflow", "path": "Automation Anywhere\\My Workflow" }
]
```

Parameter Description

Parameter	Description
name	Name of the directory/folder
path	Directory/folder path

2. Fetch list of sub-folders of a root-folder from 10.x Control Room

This API allows you to fetch the list of sub-folders for a given root-folder available in the 10.x Control Room Repository. This will help you understand the folder structure of the source Control Room.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch object details by id using the endpoint **http(s)://<host-name:port>/v1/migration/legacyrepository/childDirectories** followed by the **path** parameter
 - For example,


```
https://crdevenv.com:81/v1/migration/legacyrepository/childDirectories?path=Automation Anywhere\\My Docs
```
3. Click Send
4. The object details are listed successfully when the response status is **200**
5. The details are shown in the Body data:


```
{
  "folders":
  [
    { "name": "Log-Files", "path": "Automation Anywhere\\My Docs\\Log-Files" }
  ]
}
```

Parameter Description

Parameter	Description
folders	List of sub directories
name	Name of the directory/folder
path	Directory/folder path

3. Fetch list of files in given folder

This API allows you to fetch the list of files available in a given folder in the source Control Room repository.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch object details by id using the endpoint **http(s)://<host-name:port>/v1/migration/legacyrepository/childFiles** followed by the **path** and **excludeMetaBot** parameters
 - For example,


```
https://crdevenv.com:81/v1/migration/legacyrepository/childFiles?path=Automation Anywhere\\My Docs\\Log-Files&excludeMetaBot=false
```
3. Click Send
4. The object details are listed successfully when the response status is **200**

5. The details are shown in the Body data:

```
{
  "files":
  [
    { "id": "280", "name": "ActiveMQServer-2018-Jul-17-2.log.zip", "path": "Automation Anywhere\\My Docs\\Log-Files\\ActiveMQServer-2018-Jul-17-2.log.zip" },
    { "id": "281", "name": "IgniteServer-2018-Jul-17-4.log.zip", "path": "Automation Anywhere\\My Docs\\Log-Files\\IgniteServer-2018-Jul-17-4.log.zip" },
    { "id": "283", "name": "WebCR_Ignite-2018-Jul-17-4.log.zip", "path": "Automation Anywhere\\My Docs\\Log-Files\\WebCR_Ignite-2018-Jul-17-4.log.zip" },
    { "id": "284", "name": "WebCR_License-2018-Jul-17-4.log.zip", "path": "Automation Anywhere\\My Docs\\Log-Files\\WebCR_License-2018-Jul-17-4.log.zip" },
    { "id": "292", "name": "WebCR_Migration-2018-Jul-17-4.log", "path": "Automation Anywhere\\My Docs\\Log-Files\\WebCR_Migration-2018-Jul-17-4.log" },
    { "id": "285", "name": "WebCR_Migration-2018-Jul-17-4.log.zip", "path": "Automation Anywhere\\My Docs\\Log-Files\\WebCR_Migration-2018-Jul-17-4.log.zip" },
    { "id": "293", "name": "WebCR_Migration-2018-Jul-17-4.txt", "path": "Automation Anywhere\\My Docs\\Log-Files\\WebCR_Migration-2018-Jul-17-4.txt" }
  ]
}
```

Parameter Description

Parameter	Description
files	List of sub files
id	File id of the bot
name	Name of the directory/folder
path	Directory/folder path

4. Search for a folder by name in Control Room 10.x

This API allows you to search for a folder by given name from the source Control Room **My Docs** repository.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the GET method to fetch object details by id using the endpoint **http(s)://<host-name:port>/v1/migration/legacyrepository/folders** followed by the **taskName** parameter

For example,

<https://crdevenv.com:81/v1/migration/legacyrepository/folders?taskName=Import-Table>

3. Click Send
4. The object details are listed successfully when the response status is 200
5. The details are shown in the Body data:

```
{
  "paths":
  [ "Automation Anywhere\\My Docs\\Import-Table" ]
}
```

Parameter Description

Parameter	Description
paths	List of Directory/folder path

5. Fetch list of files for a given folder in Control Room 10.x

This API allows you to fetch a list of files from a given folder from the source Control Room **My Docs** repository.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the POST method to fetch list of files for a given folder using the endpoint **http(s)://<hostname:port>/v1/legacyrepository/files**

• **For example,**

```
https://crdevenv.com:81/v1/legacyrepository/files
```

3. Provide the list of folder paths as request payload in Body

```
[
  "string"
]
```

• **For example,** the following lists the files available

```
[
  "Automation Anywhere\\My Docs\\Import-Table"
]
```

4. Click Send
5. The object details are listed successfully when the response status is **200**
6. The Response details are shown in the Body data:

```
{
  "files":
  [
    { "id": 1281, "type": "BOT", "sourceId": "1281", "targetId": 0, "name": "Automation Anywhere\\My Doc-
s\\Import-Table\\Import-Table.txt", "status": "SUCCESS", "reason": "" },
    { "id": 293, "type": "BOT", "sourceId": "293", "targetId": 0, "name": "Automation Anywhere\\My Doc-
s\\Import-Table\\WebCR_Migration-2018-Jul-17-4.txt", "status": "SUCCESS", "reason": "" }
  ]
}
```

Parameter Description

Parameter	Description
files	List of sub files
type	file type
sourceId	Id of entity in the source database
targetId	Id of entity after migration in the target database
name	Name of the directory/folder
status	Status of response - SUCCESS, SKIPPED, or FAILED
reason	Description for the reason of status FAILED or SKIPPED

B. Migrate new or modified bots from 10.x since the last migration in 11.x

This API allows you to fetch list of bots that are new or modified in source Control Room version 10.x after data has already been migrated to destination Control Room version 11.2 . Essentially, this API allows you the liberty to continue using your 10.x environment whilst the 11.2 environment is ready to go into production.

1. Provide the "X-Authorization" parameters in Headers.
2. Use the POST method to fetch object details by id using the endpoint **http(s)://<hostname:port>/v1/legacyrepository/changedfiles**

• **For example,**

```
https://crdevenv.com:81/v1/legacyrepository/changedfiles
```

3. Provide the list of folder paths as request payload in Body

```
{
  "changeSince": "<last migration date and time>"
}
```

• **For example,** the following lists the bot names that were update post migration

```
{
```

```
"changeSince": "2018-06-25T12:05:00+05:30"
}
```

Tip: Do not specify the **changeSince** parameter to consider the delta for last migration date and time.

4. Click Send
5. The object details are listed successfully when the response status is **200**
6. The Response details are shown in the Body data:

```
{
  "changedfiles":
  [
    { "type": "BOT", "sourceId": "6", "name": "Automation Anywhere\\My Tasks\\Sample Tasks\\Import-Table.atmx" },
    { "type": "BOT", "sourceId": "7", "name": "Automation Anywhere\\My Tasks\\Sample Tasks\\List-Variable.atmx" }
  ]
}
```

Parameter Description

Parameter	Description
changedFiles	List of entities that were changed or are new since last migration run
type	entity type
sourceId	Id of entity in the source database
name	Name of the directory/folder

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200	Successful operation	NA
400	Bad request	Retry with valid parameters
401	Authentication required	Retry by providing authentication parameters
403	Unauthorized access	Ensure you have appropriate permissions to perform this operation
404	Not found	Ensure the requested data is present in Control Room
409	Conflict	Ensure the parameters provided are correct
500	Internal server error	Ensure the server is up and running
501	Permission error	Ensure that you have the required permission

Audit Logs

The **Audit Log** displays individual entry for each entity that is migrated.

DASHBOARDS

ACTIVITY

BOTS

DEVICES

WORKLOAD

AUDIT LOG

ADMINISTRATION

Control Room

ellie.brown

Audit log

Audit log

Time filter: Last 30 days

Action type Choose action type

Action type: Migration started Action type: Migration finished

Actions (18 of 105)

	STATUS	TIME	ACTION TYPE	ITEM NAME	ACTION TAKEN BY	SOURCE DEVICE	SOURCE	REQUEST ID
<input type="checkbox"/>	Successful	16:56:14 IST 2018-07-17	Migration finished	2018.07.17.16.55.59.ellie.bro...	ellie.brown	automationanywhere	Control Room	20b9da48-612e-4349-...
<input type="checkbox"/>	Successful	16:55:59 IST 2018-07-17	Migration started	2018.07.17.16.55.59.ellie.bro...	ellie.brown	automationanywhere	Control Room	d57f3d23-f9ba-450a-b...
<input type="checkbox"/>	Successful	16:13:48 IST 2018-07-17	Migration finished	2018.07.17.16.13.48.ellie.bro...	ellie.brown	automationanywhere	Control Room	f9347da5-a428-4572-a...
<input type="checkbox"/>	Successful	16:13:48 IST 2018-07-17	Migration started	2018.07.17.16.13.48.ellie.bro...	ellie.brown	automationanywhere	Control Room	c37e6689-67cd-4383-...
<input type="checkbox"/>	Successful	15:30:27 IST 2018-07-17	Migration finished	2018.07.17.15.30.25.ellie.bro...	ellie.brown	automationanywhere	Control Room	1cf4fbee-6dc4-4a35-a...
<input type="checkbox"/>	Successful	15:30:26 IST 2018-07-17	Migration started	2018.07.17.15.30.25.ellie.bro...	ellie.brown	automationanywhere	Control Room	d5a84fcd-a134-4cde-b...
<input type="checkbox"/>	Successful	13:53:10 IST 2018-07-13	Migration finished	2018.07.13.13.53.10.ellie.bro...	ellie.brown	automationanywhere	Control Room	23990981-ac19-4b35-...

When the migration process is initiated, a **Migration started** entry is logged in Audit log. Similarly when the migration process is completed, a **Migration finished** entry is logged. Between these two entries, migration entries are logged for each entity that is migrated such as **Create, Update, or Upload** operation.

- Click to view details of the process. For example, the following illustrates details of a successful migration:

Audit log > View action

Migration finished

< Back

ACTION DETAILS

Status

Successful

Action taken by

ellie.brown

Object type

Action

Source device

automationanywhere

Request ID

20b9da48-612e-4349-ba6b-d8d8a2ade3d3

Item name

2018.07.17.16.55.59.ellie.brown

Time

16:56:14 IST
2018-07-17

Action type

Migration finished

Source

Control Room

12. API for deploying and monitoring bot progress

As a Control Room administrator or a user with **View and Manage Scheduled Activity** permission, deploy bots and monitor its progress using a set of Control Room APIs.

The Deploy and Monitor Bot APIs allow you to,

1. Retrieve details of a given bot from the server repository to identify its file id to be used for bot deployment
2. Fetch list of devices (bot runners) available for automation and its automation status
3. Deploy a bot on given device(s) and fetch its automation id
4. Monitor the bot progress based on automation id

Note: The examples provided in this article are for reference only.

Before accessing the Deploy and Monitor API's you must first use the authentication API and pass it as a token to use a particular API.

1. Use the POST method to generate a token using the end point **http(s)://<hostname:port>/v1/authentication**. For this provide the Control Room instance as **Server Name /Hostname /IP** and the **Port** number.

• For example,

`https://crdevenv.com:81/v1/authentication`

2. Provide the following request payload in Headers

"X-Authorization" : "Authorization token"

"Content-Type" : "application/json"

3. Provide the following request payload in Body:

```
{
  "username": "<Username>",
  "password": "<Password>"
}
```

• For example,

```
{
  "username": "Ellie.Brown",
  "password": "12345678"
}
```

API to fetch bot details

Use this API to fetch details of a bot from the server repository. The bot id fetched from this API is used in the API for Bot Deployment.

API end point

Use the following end point to access the APIs:

<Control Room URL>/v2/repository/file/list

For example,

`https://crdevenv.com:81/v2/repository/file/list`

Fetch bots details

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to provide the following request payload in Body:

```
{ "filter
  ".
  {
    "operator": "<eq, ne, lt, le, gt, or ge>", "value": "<bot name>", "field": "fileName"
  }
}
```

Tip: Use 'eq' or 'substring' operator with field as 'name' and bot name in the 'value' of the filter, to fetch bot details of all the bots matching the bot name provided. Additional filtering, ordering and pagination rules can also be requested.

- For example, the following fetches details of the bot **Import-Table.atmx**:

```
{
  "filter":
  {
    "operator": "eq", "value": "Import-Table.atmx", "field": "fileName"
  }
}
```

3. Click **Send**

4. The action is successful when the response status is 200

5. You can view the response in the Body data.

```
{
  "page": { "offset": 0, "total": 1, "totalFilter": 1 },
  "list":
  [
    { "id": "10", "parentid": "9", "name": "Import-Table.atmx", "canDelete": true, "canDownload": true,
      "canExecute": true, "canUpload": true, "canRun": true, "lastModified": "2018-07-18T04:42:05Z",
      "lastModifiedBy": "0", "path": "Automation Anywhere\\My Tasks\\Sample Tasks\\Import-Table.atmx", "directory": false, "size": 418719, "isLocked": false, "productionVersion": "", "lockedBy": "",
      "latestVersion": "", "fileLastModified": "2018-07-18T04:42:05Z" }
  ]
}
```

Parameter Description

Parameter	Description
operator	<ul style="list-style-type: none"> NONE lt - less than le - less than equal to eq - equal to ne - not equal to ge - greater than equal to gt - greater than substring and or not
value	Name of the bot or the date
field	Name of the bot for which details are sought / lastModified / updatedOn
id	Id of the bot
parentId	Parent directory Id of the bot
name	Name of the bot
canDelete	Logged in user has rights to delete the bot - true or false
canDownload	Logged in user has rights to download the bot - true or false
canExecute	Logged in user has rights to execute the bot [#] - true or false
canUpload	Logged in user has rights to upload the bot - true or false
canRun	Logged in user has rights to run/schedule the bot - true or false
lastModified	Date and time when the bot was last updated

lastModifiedBy	The id of the user who has last modified the entity
path	Relative path of the bot
directory	Flag for directory - true or false
size	Size of the bot in KB
isLocked	Is the bot checked out by another user* - true or false
productionVersion	Current production version of the bot*
lockedBy	user id who has locked the bot *
latestVersion	Latest version of the bot*
fileLastModified	Date and time when the bot was last updated

applicable only for Meta Bots

* applicable when version control is enabled

API to fetch list of available devices (bot runner clients)

Use this API to fetch the list of devices available for automation deployment and its current status of automation. To achieve this, you need to follow the workflow mentioned here:

1. Fetch the list of devices, its type and connection status
2. Verify status of bot execution of devices

1. Fetch list of devices

Use this API to fetch the list of devices i.e. bot runners available for bot deployment. The device id retrieved from this can be passed as a parameter while using the Deployment API.

API end point

Use the following end point to access the APIs:

<Control Room URL>/v2/devices/list

For example,

<https://crdevenv.com:81/v2/devices/list>

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to provide the following request payload in Body:

```
{ "filter": {
  {
    "operator": "<eq, ne, lt, le, gt, or ge>",
    "value": "<connected or disconnected>",
    "field": "<status>" },
  "page": { }
}
```

Tip: Use 'id' and 'hostName' field for filtering on bot type and status. Additional filtering, ordering and pagination rules can also be requested.

- For example, the following fetches the status of devices of the type bot runner that are connected:

```
{
  "filter": {
    { "operator": "eq", "value": "CONNECTED", "field": "status" },
    "page": { }
  }
}
```

3. Click **Send**
4. The action is successful when the response status is 200
5. You can view the response in the Body data.

```
{
  "page": {
```

```
{ "offset": 0, "total": 2, "totalFilter": 2 },
"list":
[
{ "id": "1", "type": "BOT_RUNNER", "hostname": "CRDEVENV", "userid": "3", "username": "amy.cheng",
"status": "CONNECTED", "poolname": "" },
{ "id": "2", "type": "BOT_RUNNER", "hostname": "CRDEVENV", "userid": "5", "username": "jane.smith",
"status": "CONNECTED", "poolname": "" }
]
}
```

Parameter Description

Parameter	Description
operator	<ul style="list-style-type: none"> ┆ NONE ┆ lt - less than ┆ le - less than equal to ┆ eq - equal to ┆ ne - not equal to ┆ ge - greater than equal to ┆ gt - greater than ┆ substring ┆ and ┆ or ┆ not
value	Device status - OFFLINE, CONNECTED, or DISCONNECTED
field	Status of the device for which details are sought
id	Id of the device
type	Device type - BOT_RUNNER or BOT_CREATOR
hostName	Host name of the machine where the device is logged in
userid	User id of the device
username	User name of the device
status	Status of the device - OFFLINE, CONNECTED, or DISCONNECTED
poolname	Name of the device pool

2. API to fetch automation status

Use this API to fetch the current status of automation - whether a bot is running or not.

API end point

Use the following end point to access the APIs:

<Control Room URL>/v2/activity/list

For example,

https://crdevenv.com:81/v2/activity/list

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to provide the following request payload in Body:

```
{ "filter
":
{ "operator": "<and/or>", "operands":
[
{ "operator": "<eq, ne, lt, le, gt, or ge>", "value": "<id>", "field": "deviceid" },
{ "operator": "<and/or>", "operands":
[
```



```
{ "operator": "<eq, ne, lt, le, gt, or ge>", "value": "<Automation status>", "field": "status" },
{ "operator": "<eq, ne, lt, le, gt, or ge>", "value": "<Automation status>", "field": "status" }
}
}
}
}
```

- For example, the following fetches the status of bot execution for the device with id 2:

```
{
  "filter": {
    "operator": "and", "operands": [
      { "operator": "eq", "value": "2", "field": "deviceId" },
      { "operator": "or", "operands": [
        { "operator": "eq", "value": "RUNNING", "field": "status" }
      ]
    }
  ]
}
```

3. Click Send

4. The action is successful when the response status is 200

5. You can view the response in the Body data.

```
{
  "page": {
    "offset": 0, "total": 3, "totalFilter": 1 },
  "list": [
    { "id": "6e312e83-4115-4861-b118-26660b2b7b08", "automationName": "Import-Table_18.07.24.16.13.52_ellie.brown_API", "fileName": "Import-Table.atmx", "filePath": "Automation Anywhere\\My Tasks\\Sample Tasks\\Import-Table.atmx", "type": "TASK", "startDateTime": "2018-07-24T10:43:59Z", "endDateTime": "2018-07-24T10:44:25Z", "command": "Web Recorder", "status": "RUNNING", "progress": 43, "automationId": "6", "userId": "5", "deviceId": "2", "currentLine": 7, "totalLines": 16, "fileId": "10", "modifiedBy": "5", "createdBy": "1", "modifiedOn": "2018-07-24T10:44:26.209Z", "createdOn": "2018-07-24T10:43:52.808Z", "deploymentId": "e11d7888-1187-4ce7-b9c4-5790715bf93b", "queueName": "", "queueId": "", "usingRdp": false, "message": "Task is stopped by user.\r\n An error occurred at line number 7 of Task 'Import-Table'. Open the Task in Workbench to view action at line number 7.", "canManage": true }
  ]
}
```

Parameter Description

Parameter	Description
operator	<ul style="list-style-type: none"> NONE lt - less than le - less than equal to eq - equal to ne - not equal to ge - greater than equal to gt - greater than substring and or not
value	Automation status - DEPLOYED, RUNNING, RUN_PAUSED, UNKNOWN, COMPLETED, RUN_FAILED, RUN_ABORTED, RUN_TIMED_OUT, or DEPLOY_FAILED

field	Status of the bot for which details are sought
id	Unique execution id
automationName	Name of the automation to which the execution is associated
fileName	Name of the bot that is deployed to execute
filePath	Relative path of the bot that is deployed to execute
type	Type of activity - TASK
startDateTime	Date and time when the execution started
endDateTime	Date and time when the execution ended
command	Current command of the execution
status	Current status of the execution - DEPLOYED, RUNNING, RUN_PAUSED, UNKNOWN, COMPLETED, RUN_FAILED, RUN_ABORTED, RUN_TIMED_OUT, or DEPLOY_FAILED
progress	Current progress of the execution in percentage
automationId	Id of the automation to which the execution is associated
userId	Corresponding User id of the device where the bot is deployed
deviceId	Device id where the bot is deployed
currentLine	Current line of bot that is deployed to execute
totalLines	Total lines of the bot that is deployed to execute
fileId	Id of the bot that is deployed to execute
modifiedBy	Id of the user who last updated the execution
createdBy	Id of the user who created the automation associated with the execution
modifiedOn	Date and time when the execution was last updated
createdOn	Date and time when the execution was created
deploymentId	Deployment id to which the execution is associated
queueName	Queue name
queueId	Queue id
usingRdp	Whether the bot should be deployed using remote desktop process - true or false
message	Message if available
canManage	Does the current logged in user has rights to manage the execution - true or false

API to deploy bot

Use this to deploy a bot using device id on one or more devices using the file id of a bot.

API end point

Use the following end point to access the APIs:

<Control Room URL>/v2/automations/deploy

For example,

`https://crdevenv.com:81/v2/automations/deploy`

1. Provide the "X-Authorization" and "Content Type" parameters in Headers.
2. Use the POST method to provide the following request payload in Body:


```
{
  "fileId": "<file id of bot>",
  "deviceIds": [ "<device id 1>", "<device id 2>", "<device id 3>".... ],
  "runWithRdp": "<true or false>"
}
```

- For example, the following deploys a bot with id 10 on devices with id 1, 2, and 3 and deploys a bot using RDP:

```
{
  "fileId": "10",
  "deviceIds": [ "1", "2", "3" ],
  "runWithRDP": true
}
```

3. Click **Send**

4. The action is successful when the response status is 200

5. You can view the response in the Body data.

```
{
  "automationId": "6"
}
```

Parameter Description

Parameter	Description
fileId	ID of the bot for which is to be deployed
deviceId	Device id where the bot is to be deployed
runWithRDP	Whether the bot should be deployed using remote desktop process - true or false
automationId	Id of the automation to which the execution is associated

API to monitor bot execution

Use this API to monitor the bot progress based on automationId that is fetched using the Bot Deployment API.

Note: This API is also used to fetch a list of currently running bots on devices.

API end point

Use the following end point to access the APIs:

<Control Room URL>/v2/activity/list

For example,

<https://crdevenv.com:81/v2/activity/list>

- Provide the "X-Authorization" and "Content Type" parameters in Headers.
- Use the POST method to provide the following request payload in Body:

```
{ "filter
  ":
  {
    "operator": "<eq, ne, lt, le, gt, or ge>",
    "value": "<automationId>",
    "field": "automationId"
  }
}
```

- For example, the following fetches the status of automation with Id **15**:

```
{
  "filter":
  {
    "operator": "eq",
    "value": "6",
    "field": "automationId"
  }
}
```

3. Click **Send**

4. The action is successful when the response status is 200

5. You can view the response in the Body data.

```
{
  "page":
  { "offset": 0, "total": 3, "totalFilter": 1 },
  "list":
  [
    { "id": "6e312e83-4115-4861-b118-26660b2b7b08", "automationName": "Import-Table_18.07.24.16.13.52_ellie.brown_API", "fileName": "Import-Table.atmx", "filePath": "Automation Anywhere\\My Tasks\\Sample Tasks\\Import-Table.atmx", "type": "TASK", "startDateTime": "2018-07-24T10:43:59Z", "endDateTime": "2018-07-24T10:44:25Z", "command": "Web Recorder", "status": "RUN_PAUSED", "progress": 43, "automationId": "6", "userId": "5", "deviceId": "2", "currentLine": 7, "totalLines": 16, "fileId": "10", "modifiedBy": "5", "createdBy": "1", "modifiedOn": "2018-07-24T10:44:26.209Z", "createdOn": "2018-07-24T10:43:52.808Z", "deploymentId": "e11d7888-1187-4ce7-b9c4-5790715bf93b", "queueName": "", "queueId": "", "usingRdp": false, "message": "Task is stopped by user.\r\n An error occurred at line number 7 of Task 'Import-Table'. Open the Task in Workbench to view action at line number 7.", "canManage": true }
  ]
}
```

Note: If activity is “RUNNING” or “PAUSED FOR INPUT” status, the progress command line, currentLine and totalLines values will be shown 0.

Parameter Description

Parameter	Description
operator	<ul style="list-style-type: none"> ▪ NONE ▪ lt - less than ▪ le - less than equal to ▪ eq - equal to ▪ ne - not equal to ▪ ge - greater than equal to ▪ gt - greater than ▪ substring ▪ and ▪ or ▪ not
Value	Automation status - DEPLOYED, RUNNING, RUN_PAUSED, UNKNOWN, COMPLETED, RUN_FAILED, RUN_ABORTED, RUN_TIMED_OUT, or DEPLOY_FAILED
FileId	Status of the bot for which details are sought
Id	Unique execution id
automationName	Name of the automation to which the execution is associated
fileName	Name of the bot that is deployed to execute
filePath	Relative path of the bot that is deployed to execute
Type	Type of activity - TASK
startDateTime	Date and time when the execution started
endDateTime	Date and time when the execution ended
command	Current command of the execution
Status	Current status of the execution - DEPLOYED, RUNNING, RUN_PAUSED, UNKNOWN, COMPLETED, RUN_FAILED, RUN_ABORTED, RUN_TIMED_OUT, or DEPLOY_FAILED
progress	Current progress of the execution in percentage
automationId	Id of the automation to which the execution is associated
userId	Corresponding User id of the device where the bot is deployed
deviceId	Device id where the bot is deployed

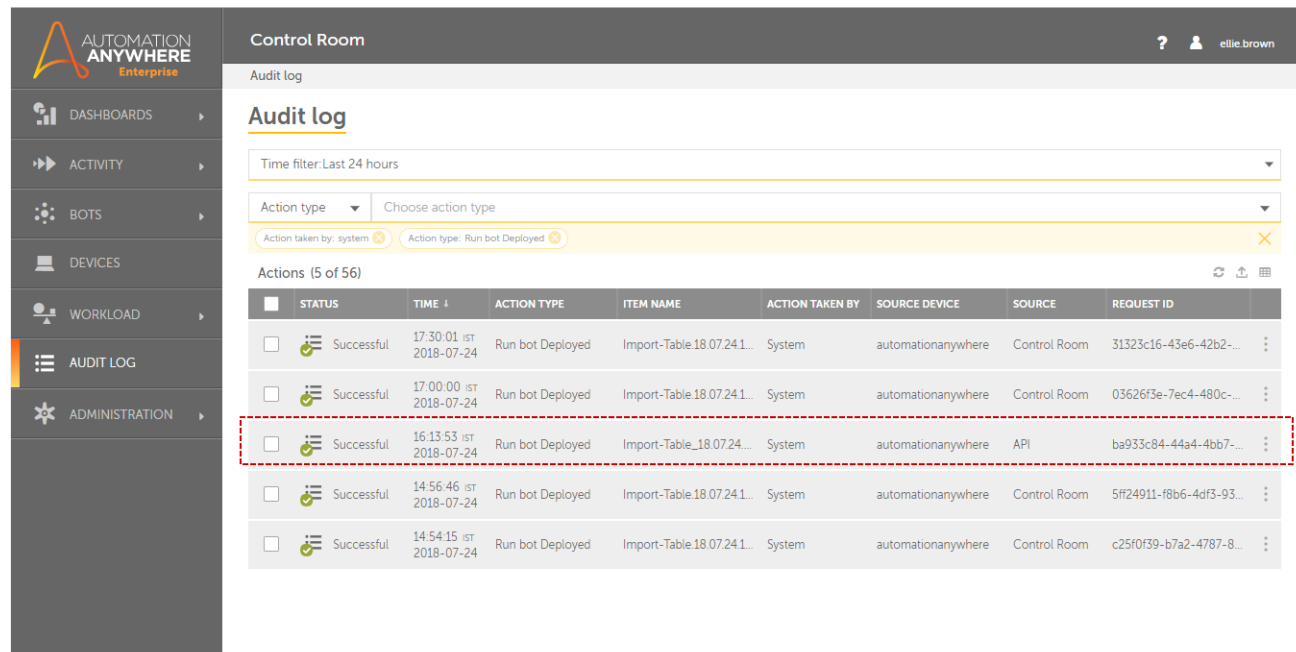
currentLine	Current line of bot that is deployed to execute
totalLines	Total lines of the bot that is deployed to execute
fileId	Id of the bot that is deployed to execute
modifiedBy	Id of the user who last updated the execution
createdBy	Id of the user who created the automation associated with the execution
modifiedOn	Date and time when the execution was last updated
createdOn	Date and time when the execution was created
deploymentId	Deployment id to which the execution is associated
queueName	Queue name
queueId	Queue id
usingRdp	Whether the bot should be deployed using remote desktop process - true or false
message	Message if available
canManage	Does the current logged in user has rights to manage the execution - true or false

API Response Codes

Http(s) Status code	Response - Description	Corrective Action
200/204	Action completed successfully	NA
304	No changes applied	Update as required
400	Bad request parameter	Retry with valid parameters
401	Authentication required	Provide authentication parameters. For example, X-Authorization key
403	Unauthorized access	Ensure you are authorized to access the Control Room
404	Not found	Ensure that the request payload is available in Control Room
409	Conflict	Ensure the parameters provided are correct
500	Internal server error	Ensure the server is up and running
501	Permission error	Ensure that you have the required permission

Audit Logs

The deployment activity using the API is logged in the Control Room Audit Log page. As a Control Room administrator or a user with View everyone's audit log actions, you can view the audit entries as shown:



Control Room

Audit log

Audit log

Time filter: Last 24 hours

Action type: Choose action type

Action taken by: system Action type: Run bot Deployed

Actions (5 of 56)

	STATUS	TIME	ACTION TYPE	ITEM NAME	ACTION TAKEN BY	SOURCE DEVICE	SOURCE	REQUEST ID
<input type="checkbox"/>	Successful	17:30:01 IST 2018-07-24	Run bot Deployed	Import-Table.18.07.24.1...	System	automationanywhere	Control Room	31323c16-43e6-42b2-...
<input type="checkbox"/>	Successful	17:00:00 IST 2018-07-24	Run bot Deployed	Import-Table.18.07.24.1...	System	automationanywhere	Control Room	03626f3e-7ec4-480c-...
<input type="checkbox"/>	Successful	16:13:53 IST 2018-07-24	Run bot Deployed	Import-Table_18.07.24.1...	System	automationanywhere	API	ba933c84-44a4-4bb7-...
<input type="checkbox"/>	Successful	14:56:46 IST 2018-07-24	Run bot Deployed	Import-Table.18.07.24.1...	System	automationanywhere	Control Room	5ff24911-f8b6-4df3-93...
<input type="checkbox"/>	Successful	14:54:15 IST 2018-07-24	Run bot Deployed	Import-Table.18.07.24.1...	System	automationanywhere	Control Room	c25f0f39-b7a2-4787-8...

The following illustration is that of a sample audit log details page for deploying a bot using Deployment API:

Audit log > View action

Run bot Deployed [< Back](#)

ACTION DETAILS

Status	Successful	Item name	Import-Table_18.07.24.16.13.52_ellie.brown_API
Action taken by	System	Time	16:13:53 IST 2018-07-24
Object type	Action	Action type	Run bot Deployed
Source device	automationanywhere	Source	API
Request ID	ba933c84-44a4-4bb7-91f8-ad5ab7ebd099		

RUN BOT DEPLOYED DETAILS

ATTRIBUTE	VALUE
Automation name	Import-Table_18.07.24.16.13.52_ellie.brow...
Bot	Import-Table.atmx
Device	CRDEVENV
Username	jane.smith
Started on	2018-07-24 16:13:52 IST
Schedule Type	N/A