AutoPass License Server

API Document

Legal Notices

Warranty

The only warranties for Hewlett Packard Enterprise Development LP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.
Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.
UNIX® is a registered trademark of The Open Group.
This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

About this PDF Version of User Guide

This document is a PDF version of the User Guide. This PDF file is provided so you can easily print multiple topics from the help information or read the User Guide in PDF format. Because this content was originally created to be viewed as User Guide in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the User Guide.

Table of Contents

Introduction

The AutoPass License Server (APLS) APIs allows developer to perform few operations such as install licenses, delete license and get features details with a product and user to perform few restricted operations on license usage report and reservation pool members.

APLS APIs are built using REST principles and follows HTTP rules, enabling HTTP clients to interact with the APIs.

Every resource is exposed as URL. The URL of each resource can be obtained by accessing the API Root Endpoints.

https://<hostname>:<port>/autopass/services/v1 for Fetch License, Get Lock, Install Licenses, Delete Licenses and Get Feature Details APIs.

https://<hostname>:<port>/autopass/wsservices/v9.3 for other APIs.

Note: the URL configuration will modify based on the APLS server deployment configurations.

Default is protocol and port is *https* and *5814.*

Getting Started

All APLS APIs requires a minimum of one mandatory header.

- **APAUTHTOKEN**: IDM Authentication Token.

For Example (Linux):

```
$ curl https://iwfvm07739.hpeswlab.net:5814/autopass/services/v1/license?productUniqueId="HPOMI_10.
11"
    -H 'APAUTHTOKEN: eyJ0eXAiOiJKV1MiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI4ZmYyNzI2NzU3ZTUyOTI5MDE1N2U1M
jk0MTBiMDA1MyIsInJlZnJlc2hfdG9rZW4iOiJTTVE2M29OWDBXNkdCcy9FUGo1M20zQlRZMFZNU3JhelhiM2NsR2xDaVU5cHN4
NmZrUzlaNU9pbUxJczRLc0hmanRtUWpWWnpVc1plS1JlTUt3OEwreG5CcXU0SC9ySy83NUE2cUVabWg4TDFvVjZXRm45Z2FVUTB
EdGhTdWoxRGJEQmVaVlpLcE5hUzdtRkxpazFDUXBrZ1ZYWEJBYWh0a01wdVFtN0xEN0s3aVJQZGJ1b0diWjdKSUZONW9Hd1ZVUD
R1L0dWOFNxaHRaQ2IzbHVUenZON0dYMzc2R0xxbHY1MEdSRmJ4NU1taUpNUnVPaVRXcm1zMGVOMjQ2TENNN05nNjBDeHRZc2thO
XAyVXNYSmJWd2prNzVJL1Y0cVkwcEJ5bnJmTnVzbUVReGp4a3hkQlR4Q2hON0JSWnU2QU03WGZJUW1NRlNQbmhNbnRjK2JEWEFO
disrbXpCTmJHdTBpQkxKMlY3WHI4VXN6aWVWODZhazMyTmpYWWR0WFFoUy9zSWVqaTlId0hNRFNjNHVTbXBrOWNxSnI2aHl3L3V
sUVhpajJGR1BmTk15a0RSVGlESGd3M3R6RmpKYmJNeCtBcmg2cEFPUHNQcHE2eTBYSzdtZFFnTml2aC9GMkIiLCJpc3MiOiJJZE
0gMS4xMC40IiwiY29tLmhwZS5pZG06dHJ1c3RvciI6bnVsbCwiZXhwIjoxNDgwMzQzNTY2LCJjb20uaHAuY2xvdWQ6dGVuYW50I
jp7ImlkIjoiOGZmMjcyNjc1N2U1MjkyOTAxNTdlNTI5M2E0NTAwMTIiLCJuYW1lIjoiUHJvdmlkZXIiLCJkZXNjcmlwdGlvbiI6
IlByb3ZpZGVyIiwiZW5hYmxlZCI6dHJ1ZX0sInBybiI6ImFkbWluIiwiaWF0IjoxNDgwMzQxNzY2LCJqdGkiOiJlNWJmYzVjYy0
3NWRkLTQxNGMtYmIwYi1mNzVhODdiOTkyZWUifQ.BREVdpJdSMtQRuEQYTHm_RxDLoBhMFLVGmD6ZyGUmWA'
```

Authentication

Most AutoPass License Server API need to be authenticated using an IDM authentication token.
To obtain an IDM authentication token programmatically using the IDM Rest API.
Submit an HTTPS POST request to the below URL.

```
$ curl https://<IDM_HostName>:<IDM_PORT>/idm-service/v2.0/tokens
      -H 'Authorization: Basic aWRtVHJhbnNwb3J0VXNlcjppZG1UcmFuc3BvcnRVc2Vy'
```

The POST body should include the request content in the below format,

```
{
  "passwordCredentials" : {
    "username" : "admin",
    "password" : "!qaz2wsx"
  },
  "tenantName" : "Provider"
```

```
}
```

Please note that you need pass IDM's base authentication credentials in the **header** to get a valid IDM token.

- **Authorization**: Authentication request header.

For Example (Linux):

```
$ curl https://iwfvm07739.hpeswlab.net:5814/autopass/wsservices/v9.3/reservation/pool/members?poolName=”Development”
    -H 'Authorization: Basic YXBsc1VzZXI6cGFzc3dvcmQ='
```

Authentication

The APIs use “Basic” authentication to authenticate and authorize users to perform operations on APLS APIs.

Default view only user configured in APLS is **‘aplsUser’: “password”** (*YXBsc1VzZXI6cGFzc3dvcmQ=)*

To manage users in APLS refer **“User Management”** section in user guide.

To pass the credentials to the rest API use the below format:

<user name>:<password>

Encode the above value using *Base64* encoding (e.g., *YXBsc1VzZXI6cGFzc3dvcmQ=*)

Append encoded value with “Basic“, e.g. “Basic *YXBsc1VzZXI6cGFzc3dvcmQ=”*

Add above as “Authorization” header value

Points to Note

Authentication token should be generated based on the user managed in APLS. If the user is deleted, the APIs will not able to authenticate and authorize the token and you will receive *401 error code*

To add or delete operations, the authenticated token should belong to an administrator user in APLS.

In case of SSL connection issue with *curl* command, search for “Curl disable certificate verification” to allow SSL connection without validating the server certificate. Please note the curl command given above is just a sample and meant for testing purpose only. It is always recommended that client implement code to validate the server certificate before proceeding with the next step.

HTTP Methods

Using **GET** method, user can get the list of resources or details of a particular instance of a resource.

For e.g., below is to get a list of license installed in the APLS for a product

```
$ curl https://iwfvm07739.hpeswlab.net:5814/autopass/services/v1/product/features?productUniqueId="
HPE UFT_14.0"
    -H 'APAUTHTOKEN: eyJ0eXAiOiJKV1MiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI4ZmYyNzI2NzU3ZTUyOTI5MDE1N2U1M
jk0MTBiMDA1MyIsInJlZnJlc2hfdG9rZW4iOiJTTVE2M29OWDBXNkdCcy9FUGo1M20zQlRZMFZNU3JhelhiM2NsR2xDaVU5cHN4
NmZrUzlaNU9pbUxJczRLc0hmanRtUWpWWnpVc1plS1JlTUt3OEwreG5CcXU0SC9ySy83NUE2cUVabWg4TDFvVjZXRm45Z2FVUTB
EdGhTdWoxRGJEQmVaVlpLcE5hUzdtRkxpazFDUXBrZ1ZYWEJBYWh0a01wdVFtN0xEN0s3aVJQZGJ1b0diWjdKSUZONW9Hd1ZVUD
R1L0dWOFNxaHRaQ2IzbHVUenZON0dYMzc2R0xxbHY1MEdSRmJ4NU1taUpNUnVPaVRXcm1zMGVOMjQ2TENNN05nNjBDeHRZc2thO
XAyVXNYSmJWd2prNzVJL1Y0cVkwcEJ5bnJmTnVzbUVReGp4a3hkQlR4Q2hON0JSWnU2QU03WGZJUW1NRlNQbmhNbnRjK2JEWEFO
disrbXpCTmJHdTBpQkxKMlY3WHI4VXN6aWVWODZhazMyTmpYWWR0WFFoUy9zSWVqaTlId0hNRFNjNHVTbXBrOWNxSnI2aHl3L3V
sUVhpajJGR1BmTk15a0RSVGlESGd3M3R6RmpKYmJNeCtBcmg2cEFPUHNQcHE2eTBYSzdtZFFnTml2aC9GMkIiLCJpc3MiOiJJZE
0gMS4xMC40IiwiY29tLmhwZS5pZG06dHJ1c3RvciI6bnVsbCwiZXhwIjoxNDgwMzQzNTY2LCJjb20uaHAuY2xvdWQ6dGVuYW50I
jp7ImlkIjoiOGZmMjcyNjc1N2U1MjkyOTAxNTdlNTI5M2E0NTAwMTIiLCJuYW1lIjoiUHJvdmlkZXIiLCJkZXNjcmlwdGlvbiI6
IlByb3ZpZGVyIiwiZW5hYmxlZCI6dHJ1ZX0sInBybiI6ImFkbWluIiwiaWF0IjoxNDgwMzQxNzY2LCJqdGkiOiJlNWJmYzVjYy0
3NWRkLTQxNGMtYmIwYi1mNzVhODdiOTkyZWUifQ.BREVdpJdSMtQRuEQYTHm_RxDLoBhMFLVGmD6ZyGUmWA'
```

To get a list of pool member

```
$ curl https://iwfvm07739.hpeswlab.net:5814/autopass/wsservices/v9.3/reservation/pool/members?poolNa
me="Develpoment"
    -H 'Authorization: Basic YXBsc1VzZXI6cGFzc3dvcmQ='
```

Currently the below three methods are used in API queries.

Find the below table for reference:

Method	Description
GET	**To retrieve resources.**
POST	**To create resources and performing resource actions.**
DELETE	**To delete resources.**

Response

Response, by default in the XML format.

Find the below table for reference:

NODE NAME	Description
Schema	**Each response schema differs based on the resources**
Status	**Each API call will return an status with *SUCCESS/ PARTIAL SUCCESS/FAIL***
errorCode	**Custom error code from APLS server**
errorMessage	**Generic error message**
customMessage	**Custom error message specifying the nature of the failure**

Sample response structure,

```
<FeatureListResponse>
   <feature>
      <featureID>10596</featureID>
      <featureVersion>1</featureVersion>
      <featureDescription>HPE Unified Functional Testing Seat User</featureDescription>
      <activationDateInUTCSeconds>1480118400</activationDateInUTCSeconds>
      <expirationDateInUTCSeconds>900703</expirationDateInUTCSeconds>
      <totalCapacity>10</totalCapacity>
      <availableCapacity>10</availableCapacity>
   </feature>
```

```
<ReservationPoolResult>
   <name>Development</name>
   <status>SUCCESS</status>
   <members>
      <userNameList>
         <userName>ASIAPACIFIC/rpadmava</userName>
      </userNameList>
      <ipAddressList/>
      <hostList/>
      <clientIDList/>
   </members>
</ReservationPoolResult>
```

Other Format Support

All GET APIs supports *JSON* format. To return the response in JSON format Accept head as well for which the required response format need to be specified in the respective request's Accept header.

For example:

Pass **Accept** header as **application/json** below result displays

```
{"feature": [{
   "featureID": "10596",
   "featureVersion": "1",
   "featureDescription": "HPE Unified Functional Testing Seat User",
   "activationDateInUTCSeconds": "1480118400",
   "expirationDateInUTCSeconds": "900703",
   "totalCapacity": {"value": "10"},
```

```
{
   "name": "Development",
   "status": "SUCCESS",
   "members":    {
      "userNameList": {"userName": ["ASIAPACIFIC/rpadmava"]},
      "ipAddressList": {"ipAddress": []},
      "hostList": {"host": []},
      "clientIDList": {"clientID": []}
   }
}
```

Errors

APLS APIs uses HTTP status codes to indicate success or failure of an API call. In general, status codes in the *2xx* range means ‘success’, *4xx* range means there was an ‘error’ in the provided information, and those in the *5xx* range indicates ‘server side errors’.

Commonly used HTTP status codes by APLS are listed below:

Status Code	Description
200	**Ok**
201	**Created**
401	**Unauthorized (Invalid AuthToken)**
403	**Forbidden**
404	**URL Not Found**
406	**Not Acceptable**
500	**Internal Error**

Fetch License

This API can be used to get the license keys for a given product installed on APLS. To fetch all the license keys for a given product installed on APLS, the user need to pass the Unique Product Identifier of the given product as the query parameter of the HTTP GET request.

This API also has the capability to fetch license keys which are installed or deleted from APLS server after a given last server time stamp. The last server time stamp will be retuned from the APLS server in response body with each successful response from the server. This server time can be used to make subsequest requests to the APLS server to get the delta of changes after the given time stamp. The server time stamp is represented in seconds from epoch.

Attributes

Member attributes should be passed to the APLS server using HTTP GET methods as query parameters in the URL. The response is in XML media type only.

ATTRIBUTE	DESCRIPTION
productUniqueId	**UniqueIdentifier for a given product as retuned by the AutoPassJ Core.**
lastServerTimeStamp	The server time stamp represented in seconds from epoch.

Request

REQUEST

Method	URL
GET	**https://<APLM_IP_OR_HOSTNAME>:<PORT>/autopass/services/v1/license?productUniqueId=<Unique_Product_ID>&lastServerTimeStamp=<Last_Server_Response_Time_Stamp_In_Seconds>**

Response

- The response of this API will be in XML only. A sample XML response would look like this –

STATUS	RESPONSE
200 Created	**A successful response from the API. The license details are returned in response.**

```
<ReportLicenseResponse>
   <pageNumber>1</pageNumber>
   <numberOfRecordsPerPage>4</numberOfRecordsPerPage>
   <totalPages>1</totalPages>
   <validLicenseList>
      <license>
         <licenseKey keyType="OVKEY4" contentType="base64">OUJURyBBOTlBIE
g5UFEgR0hXWiBVOUI1IEhXU1YgWTlKTCBLTVBMIE5VV0MgNkRGNCA2Uk1TIEtIV0UgSjZSNiB
YRlpYIENNUkcgSFBNUiBNSDVVIEE1ODkgVUVQSyA5SzJKIENLRFUgM1hXVCBBU0xQIDRQOTkg
```

```
UFRUMiA0WDlCIFo1Q1kgWlpKRSBOVTIyIExIVFUgVkxYTCBIRk1QIEJLVEcgMlZBQiBWUzdLI
EM3TEUgTFhWNCAyTkg3IEZUS0MgMkJFVSA0TUpTIE5DWFIgRTJVWCAyTVpVIEVOVVUgQkdONS
BMOFNHIDI3RDcgWUs5WSA0RlQ5ICJPcmFjbGUgRS1CdXNpbmVzcyAtIEV2YWx1YXRpb24gRXh
wbGljaXQgZmVhdHVyZSI=</licenseKey>
         <productID>LR</productID>
         <productVersion>14.0</productVersion>
         <featureID>7176</featureID>
         <featureVersion>1</featureVersion>
      </license>
   </validLicenseList>
   <removedLicenseList>
      <license>
         <licenseKey keyType="OVKEY4" contentType="base64">OUJTRSBCOU1BIE
g5UFEgOEhYWiBVREI0IEhXU0ogWTlKTCBLTVBMIEI4OUggTVpWVSBHUjRVIEtIV0UgSjdUNyB
YRlVVIENNUkcgSFBNUiBFSDVVIEE1ODkgNFZNSyA5Q1o5IEZNNU0gN1lZQSBSU1hFIDk5VU0g
TE1NTiA0UUEyIFdXMlcgOUI5USA5M0pUIERGNFYgR0Y3QyBETUozIEJLVDggV1ZEQyBUWDlVI
EQzU0UgVlNXNiBGQzRYIDVLRzIgRjZBRCA0TUZLIFpHRUIgRVpVTiBMRU5ZIkFjdGl2ZVgvSm
F2YXNjcmlwdCAtIFRpbWUgbGltaXRlZCBFeHBsaWNpdCBmZWF0dXJlIg==</licenseKey>
         <productID>LR</productID>
         <productVersion>14.0</productVersion>
         <featureID>10421</featureID>
         <featureVersion>1</featureVersion>
      </license>
   </removedLicenseList>
   <serverTimeStamp>1481621674</serverTimeStamp>
</ReportLicenseResponse>
```

202 OK

If the product information is not available at APLS and a request is made. This another successful response from server which denotes the product information is not available at the APLS server.

```
<ReportLicenseResponse>
   <errorDetail>
      <errorCode>14008</errorCode>
      <errorMessage>Product Definition is not found for the product in th
e APLM server.</errorMessage>
      <customMessage>Product Definition is not found for the product in t
he APLM server.</customMessage>
   </errorDetail>
</ReportLicenseResponse>
```

406 Not Acceptable

If the last server timestamp format is incorrect (not in long type) then the following error is sent in response.

```
<ReportLicenseResponse>
   <errorDetail>
      <errorCode>14001</errorCode>
      <errorMessage>The format of APLM server timestamp must be long.</er
rorMessage>
      <customMessage>The format of APLM server timestamp must be long.</c
ustomMessage>
   </errorDetail>
</ReportLicenseResponse>
```

406 Not Acceptable

If the unique product identifier not passed in the request then following error is thrown in response.

```
<ReportLicenseResponse>
   <errorDetail>
      <errorCode>14009</errorCode>
      <errorMessage>Product unique identifier cannot be null or empty.</e
rrorMessage>
```

```
      <customMessage>Product unique identifier cannot be null or empty.</
customMessage>
   </errorDetail>
</ReportLicenseResponse>
```

Points to Note

- Unique Product Identifier is a MUST. This API can return licenses installed on APLS only if a valid unique product identifier is passed. AutoPass CORE has the capability to get the unique product identifier for a given product. You need to contact the AutoPass product integration team to understand how to get the unique product identifier using CORE.

- The Product Definition (PD) file of the product for which this API is to be invoked must be available on the APLS server prior to making this call. AutoPass Core version 9.4 has the capability to push the PD file to APLS automatically when AutoPass Core is running under APLMS mode.

Get Lock

This API can be used to get the Lock Value for a given product from APLS. APLS has the capability to manage two types of lock values, one lock value is the global lock value per installation of APLM and which will also be hosted on the APLS management console UI. Second set of lock values will be specific to the products installed at the APLS server.
Products can chose of work with APLS's global lock value or they can chose to have their own defined lock value to be available at APLS. If products decides to go with their own lock value for the licenses installed at APLS then they need to contact AutoPass integration team to help them in this process.

Attributes

Member attributes should be passed to the APLS server using GET, POST methods as query parameters in the URL for GET method to in the request body as XML or JSON for POST methods. Lock API uses HTTP POST method with the following attributes to be passed in the body of the request

ATTRIBUTE	DESCRIPTION
productUniqueId	**UniqueIdentifier for a given product as retuned by the AutoPassJ Core.**

Request

REQUEST	
Method	**URL**
POST	**https://<APLS_IP_OR_HOSTNAME>:<PORT>/autopass/services/v1/lock**

Sample Request

```
<LmsLockRequest>
        <productUniqueId>HP UFT_12.5</productUniqueId>
</LmsLockRequest>
```

Response

- The response of this API will be XML or JSON depending upon the Media Type set in the request. A sample XML response would look like this –

STATUS	RESPONSE
200 Created	**A successful response from server. The APLS lock code is returned in response. In multi lock configuration exists at APLS server then the pre configured lock code for the given product will be returned.**

```
<LmsLockResponse>
   <lock>
      <lockDetail lockType="DEVICE">43858EB-4A9C529</lockDetail>
   </lock>
```

```
</LmsLockResponse>
```

The “lockType” xml attribute in the reaponse signifies that the type of lock value as returned by the API, APLS support “DEVICE” lock type as of this release.

Points to Note

- For the first time when a user makes a call to the Lock service, it can pass the unique product identifier of the APLS product itself. In that case, this API will return the global lock value as hosted by the APLS server.

Install Licenses

This API can install licenses by loading a single license key file or multiple license keys in one file. You obtain license keys on a per-product and per-feature basis. For details on obtaining license keys, see the individual product documentation.
If you received a license key in the form of a file, you can invoke this micro service to install license(s) in APLS server.

Request

REQUEST

Method	URL
POST	https://<hostname>:<port>/autopass/services/v1/licenses

Request Examples/ Configure a Request

After you created a request, you may now configure it as shown on the below. There are 3 items to take note of.

1. The HTTP method is POST.

2. Request Parameter Value
   Name: file
   Value: file:<PATH_OF_LICENSE_FILE_NAME>

3. Media Type
   This must be set to “multipart/form-data”

The above request contains the media type “multipart/form-data” and file:< PATH_AND_FILE_NAME> as an input.
The API is called with the above information as an input with **HTTP POST** method to install the list of licesens.

Response

The Response by default will be in XML format. The status of this call can be identified through *status* field of the response object which will contain SUCCESS / PARTIAL SUCCESS / FAIL.
If the API request to install license(s) are successful, the licenses will be added to the specified license Management in APLS UI ->View Licenses.

STATUS	RESPONSE
200 OK	**Below is the response when the POST call to install licenses is successful.**

```
<LicenseManagementServiceResponse>
   <status>SUCCESS</status>
   <LicenseInfo>
      <productID>HP UFT</productID>
      <featureId>10618</featureId>
      <featureVersion>1</featureVersion>
      <capacity>100</capacity>
```

```
      <startTimeInUTCSeconds>1456790400</startTimeInUTCSeconds>
      <expiryTimeInUTCSeconds>900703</expiryTimeInUTCSeconds>
      <lockCode>any</lockCode>
      <LicenseString contentType="Base64" keyType="OVKEY4">WUFEQSBBOUVBIE
g5UFEgS0hWMiBVM0I0IEhXVzUgWTlKTCBLTVBMIEI4OUggTVpWVSA2UkVXIDlIV0UgSkJSSCA
3NVhKIENNUkcgSFBNUiA0R1hVIEFGRzkgNFVYUyBFSzI5IFhQVU0gN1pMUCBRS1Q1IEVQQk0g
VDhQRiBEUUJDIFNZU1ggWUJBNCBVNTJBIEw3WEEgSkg3TCBMTlFEIEpWOE0gTTQyOCBMSzRVI
FI0V0EgUjI5WCBaU01QIDdLRUMgSkdBRCBFTVZBIDlHRUIgSFpRWiBVQTRQIDdYSlUgQkxEVi
A1U0tOIFoySDciUXVpY2tUZXN0IFByb2Zlc3Npb25hbCBBZGQtaW4gZm9yIFNBUCBTb2x1dGl
vbnMgQ29uY3VycmVudCBVc2VyIg==</LicenseString>
      <licenseServiceStatus>
         <status>SUCCESS</status>
         <licenseServiceOperation>
            <operation>ADDED</operation>
         </licenseServiceOperation>
      </licenseServiceStatus>
   </LicenseInfo>
</LicenseManagementServiceResponse>
```


400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
401-UN AUTHRIZED	**In case if there is no IDM authentication token is passed,you will get 401 un authorized from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue.**

Points to Note

- It is recommended to restrict the maximum number of licenses to be added to 100, for faster response.
- Currently the request input support the media type “multipart/form-data” only. The response media can be either an xml or json based on the “Accept” header configuration in your query.
- From the above success response, you can see a license string tag which has base64 format and license key type indicates whether the license file is ovekey4/safe key. You can decode the string to verify the license.
- Also this API need to be authenticated using an IDM authentication token, please ensure you pass the “APAUTHTOKEN” for this query using the IDM authentication token.

Delete Licenses

This REST API will allow you to DELETE a list of licenses from a license management. Incase if you do not want a specific license, you can delete a license. By using this API You can delete any unused license that is displayed in the APLS View Licenses UI.
Create a HTTP DELETE request to delete the license from a given license. IDM Authentication details must be also submitted as part of HTTP request header.

- Deletion of licenses from APLS can be done in two ways: By signing in to the APLS UI. For more details how to delete license from license manager. Please refer user guide (**"Archived License (License Management Pane)"**)

- By calling the API by issuing the HTTP DELETE request to the APLMS endpoint/handler.

By calling REST API by issuing the HTTP POST request to the APLS endpoint/handler. Please read the following details for more information about how to pass POST request.

Please read the following details for more information about how to pass DELETE request.

Important information

Ensure a license(s) exists through APLMS UI in order for this rest service to successfully delete the license. Please refer the user guide (**"View Licenses (License Management Pane)"**).
The following is the End Point to be used with HTTP DELETE request. The request shall be in *XML* format.

Request

REQUEST

Method	URL
DELETE	**https://<hostname>:<port>/autopass/services/v1/licenses**

Sample Requests:

Use case remove license feature from a license management

```
<RemoveLicenseRequest>
     <RemoveLicenseInfo>
                 <featureID>10594</featureID>
                 <featureVersion>1</featureVersion>
            </RemoveLicenseInfo>
</RemoveLicenseRequest>
```

The abve request contains the media type “application/xml” as an input.
The API is called with the above xml as an input with **HTTP DELETE** method to delete the list of licenses.
You can also use combination of licenseID can be used to remove licenses.

For example:

```
<RemoveLicenseRequest>
   <RemoveLicenseInfo>
                 <featureID>10594</featureID>
                 <featureVersion>1</featureVersion>
              <LicenseIDList>
                    <LicenseID> </LicenseID>
```

```
            </LicenseIDList>
   </RemoveLicenseInfo>
</RemoveLicenseRequest>
```

The below sample request contains the media type “application/json” as an input.

For example:

```
{
   "RemoveLicenseInfo": {
      "featureID": "10618",
      "featureVersion": "1",
      "LicenseIDList": {
         "LicenseID": “”
      }
   }
}
```

Response

The Response by default will be in XML format which contain a HTTP status code and response XML. Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE

200 OK	Below is the response when the DELETE call to remove license from APLS is successful
	
	`<LicenseManagementServiceResponse>`
	`<status>SUCCESS</status>`
	`<LicenseInfo>`
	`<productID>HP UFT</productID>`
	`<featureId>10594</featureId>`
	`<featureVersion>1</featureVersion>`
	`<capacity>100</capacity>`
	`<startTimeInUTCSeconds>1456790400</startTimeInUTCSeconds>`
	`<expiryTimeInUTCSeconds>900703</expiryTimeInUTCSeconds>`
	`<lockCode>any</lockCode>`
	`<licenseServiceStatus>`
	`<status>SUCCESS</status>`
	`<licenseServiceOperation>`
	`<operation>ARCHIVED</operation>`
	`</licenseServiceOperation>`
	`</licenseServiceStatus>`
	`</LicenseInfo>`
	`</LicenseManagementServiceResponse>`

401-UN AUTHRIZED	**In case if there is no IDM authentication token is passed,you will get 401 un authorized from the server**
400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue. Or a wrong URL is configured.**

Points to Note

- The request input support the media type “application/xml or application/json”. The response media can be either an xml or json based on the “Accept” header configuration in your query.

- Also this API need to be authenticated using an IDM authentication token, please ensure you pass the “APAUTHTOKEN” for this query using the IDM authentication token.

- From the above request,the LicenseID will get it from the core “autopassj.reportLicenseSet(lock,false)” will retun the list of licenses from APLS which you can then use license.getLicenseID() to get the licenseID which is mapped to the feature ID and feature version and pass it to above request as an input to remove any specific licenses.

Get Feature Details

This API's is to get the feature details (aggregated capacity) from APLS.The following are the 3 API's which will return the feature details aggregated information based on by passing the product information (product unique ID or combination of product ID and product version) or will return for a specific feature. Please find the below request URI's.

Request

REQUEST

Method	URL
GET	**https://<IP>:5814/autopass/services/v1/product/features?productUniqueId={productcode_productfullversion}**
	https://<IP>:5814/autopass/services/v1/product/features?productId={product id}&productVersion={pd version}
	https://<IP>:5814/autopass/services/v1/product/features?featureId={feature id}&featureVersion={feature version}

@param productUniqueId:

UniqueIdentifier for a given product. Format for the value looks like productcode_productfullversion. Product code and product version values can be found in the PD file.

@param productId:
Product code of the product as identified in the Product Definition file. This field is mandatory.

@param productVersion:

Version of the product as identified in the Product Definition file. This field is mandatory.

Note: The combination of the Product Code and Product Version is unique.

@param featureId:

The specific feature number for each feature of the selected product.

@param featureVersion:

The version number of the feature (without the feature ID).

The below two query params are optional and can be used along with all the above three URLs. Both lockType and lockValue should be use together and will retrieve the feature details based on the lock values passed. By default wild carded lock values will be used to retrieve the feature details.

@param lockType

The lock type for which the feature details are to be retrieved. It is one of the following options,

- 1 – IP Address
- 4 – Device ID

@param lockValue

The lock value for which the licenses for a feature are to be retrieved. For e.g., Say lockType is 1, lockValue can be 16.12.36.3.

The API is called with the above information as an input with **HTTP GET** method to get the feature details from APLMS.

Important information

Ensure a product definition file(s) exists through APLS pdfile location folder or feature details exists through APLS view license page in order for this rest service to successfully get the aggregated details.

Response

The Response by default will be in XML format. The status of this call can be identified through *status* field of the response object which will contain SUCCESS / PARTIAL SUCCESS / FAIL.
Below is the response incase if any one of the above API's request to get feature details are successful.

STATUS	RESPONSE
200 OK	**Below is the response when the GET method is called for a product either using productUniqueId or productID/productVersion. All the features of the product with aggregated capacity is listed under FeatureListResponse**

```
<FeatureListResponse>
   <feature>
      <featureID>10526</featureID>
      <featureVersion>1</featureVersion>
      <featureDescription>HP Svc Health Rep Standard 50 SH Nodes SW</feat
ureDescription>
      <activationDateInUTCSeconds>1480579200</activationDateInUTCSeconds>
      <expirationDateInUTCSeconds>900703</expirationDateInUTCSeconds>
      <totalCapacity>100</totalCapacity>
      <availableCapacity>100</availableCapacity>
   </feature>
   <feature>
      <featureID>1053</featureID>
      <featureVersion>1</featureVersion>
      <featureDescription>HP Service Health Reporter Scalable License</fe
atureDescription>
      <activationDateInUTCSeconds>1480579200</activationDateInUTCSeconds>
      <expirationDateInUTCSeconds>900703</expirationDateInUTCSeconds>
      <totalCapacity>100</totalCapacity>
      <availableCapacity>100</availableCapacity>
   </feature>
</FeatureListResponse>
```

200 OK	**Below is the response when the GET method is called for a specifie feature using featureId and featureVersion. The aggregated capacity fo the feature is listed under FeatureResponse**

```
<FeatureResponse>
   <feature>
      <featureID>10526</featureID>
      <featureVersion>1</featureVersion>
      <featureDescription>HP Svc Health Rep Standard 50 SH Nodes SW</feat
ureDescription>
```

	<activationDateInUTCSeconds>1480579200</activationDateInUTCSeconds> <expirationDateInUTCSeconds>900703</expirationDateInUTCSeconds> <totalCapacity>100</totalCapacity> <availableCapacity>100</availableCapacity> </feature> </FeatureResponse>
406 NOT ACCEPTABLE	**Below is the response if the product is configured in APLS, however is available but license are not installed** <FeatureResponse> <status>FAIL</status> <errorDetail> <errorCode>5997</errorCode> <errorMessage>No license is found in Memory</errorMessage> <customMessage>No licenses found in license file</customMessage> </errorDetail> </FeatureResponse>
406 NOT ACCEPTABLE	**Below is the response when the GET call to feature details is fail.mean if there is no PD file or license details not present in APLS.** <FeatureResponse> <status>FAIL</status> <errorDetail> <errorCode>14008</errorCode> <errorMessage>Product Definition is not found for the product in th e APLM server.</errorMessage> <customMessage>Product Definition is not found for feature 1:1 in t he APLM server.</customMessage> </errorDetail> </FeatureResponse>
401-UN AUTHRIZED	**In case if there is no IDM authentication token is passed,you will get 401 un authorized from the server**
400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue.**

Points to Note

- The response media can be either an xml or json based on the “Accept” header configuration in your query.
- Also all the above 3 API’s need to be authenticated using an IDM authentication token, please ensure you pass the “APAUTHTOKEN” for this query using the IDM authentication token.

Get Product Feature Details

This API is used to get the list of feature details and the list of feature license key details for the requested product definition file id and product details from APLS.

Request

REQUEST

Method	URL

GET	**https://<IP>:5814/ autopass/services/v10.5/getProductFeatures?pdfId={pdf_file_id}&pdfVersion={pdf_file_version}&productId={product id}}&productVersion={pd version}**

@param pdfId:

Product Definition file identifier.

@param pdfVersion:

Product Definition file version.

@param productId:

Product code of the product as identified in the Product Definition file.

@param productVersion:

Version of the product as identified in the Product Definition file. This field is mandatory.

Note: The combination of the Product Code and Product Version is unique.

Important information

Ensure a product definition file(s) exists through APLS pdfile location folder or feature details exists through APLS view license page

Response

The Response by default will be in XML format. The status of this call can be identified through *status* field of the response object which will contain SUCCESS / PARTIAL SUCCESS / FAIL.
Below is the response for the above API request to get product feature details from the server.

STATUS	RESPONSE
200 OK	

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProductFeature>
    <Product>
        <productId>HPUFT</productId>
        <description>HP Unified Functional Testing</description>
        <pdfID>10027</pdfID>
        <pdfVersion>1.0</pdfVersion>
        <version>12.50</version>
        <releaseDate>0</releaseDate>
        <entitlementRequiredIndicator>true</entitlementRequiredIndicator>
        <ProductFeatureList>
            <Feature>
                <featureId>10624</featureId>
                <version>1</version>
                <description>QuickTest Pro Stingray Add-in Concurrent Use
r</description>
                <licenseModel>FLOATING</licenseModel>
                <isClockTamperingDetectionEnabled>true</isClockTamperingD
etectionEnabled>
                <commuterstatus>ADMIN ALLOWED</commuterstatus>
                <commuternoofdays>30</commuternoofdays>
```

```
            </Feature>
            <Feature>
                <featureId>10612</featureId>
                <version>1</version>
                <description>QuickTest Pro Siebel Add-in Concurrent User<
/description>
                <licenseModel>FLOATING</licenseModel>
                <isClockTamperingDetectionEnabled>true</isClockTamperingD
etectionEnabled>
                <commuterstatus>ADMIN ALLOWED</commuterstatus>
                <commuternoofdays>30</commuternoofdays>
            </Feature>
        </ProductFeatureList>
        <machineDetails>
            <hostname>457bdd6e30de</hostname>
            <ipAddressList>
                <ipAddresses>172.17.0.4</ipAddresses>
                <ipAddresses>127.0.0.1</ipAddresses>
                <ipAddresses>fe80:0:0:0:42:acff:fe11:4%eth0</ipAddresses>
            </ipAddressList>
        </machineDetails>
    </Product>
    <LicensedFeatureList>
        <FeatureInfo>
            <featureId>10622</featureId>
            <featureVersion>1</featureVersion>
            <featureDescription>QuickTest Pro PeopleSoft Add-in Concurren
t User</featureDescription>
            <TotalCapacity>0</TotalCapacity>
            <TotalUnits>0.0</TotalUnits>
            <AvailableCapacity>0</AvailableCapacity>
            <AvailableUnits>0.0</AvailableUnits>
            <TotalUsedCapacity>0</TotalUsedCapacity>
            <TotalUsedUnits>0.0</TotalUsedUnits>
            <StartDate>1489795200</StartDate>
            <FirstExpiryDate>900703</FirstExpiryDate>
            <LastExpiryDate>900703</LastExpiryDate>
            <licenseModelType>FLOATING</licenseModelType>
            <associatedLicenseList>
                <license>
                    <LicenseId>95c5807d-8695-42e7-9922-d1ebcf1f6ed9</Lice
nseId>
                    <rawKey>&lt;![CDATA[&lt;?xml version=&quot;1.0&quot;
encoding=&quot;UTF-8&quot; standalone=&quot;yes&quot;?&gt;
&lt;RAW_LICENSE_KEY&gt;
..
&lt;/RAW_LICENSE_KEY&gt;]]&gt;</rawKey>
                    <rawKeyContentType>XML</rawKeyContentType>
                    <keyType>SafeKey</keyType>
                    <capacity>100</capacity>
                    <creationDate>1489834150</creationDate>
                    <startDate>1489795200</startDate>
                    <endDate>900703</endDate>
                    <durationDays>-1</durationDays>
                    <validity>PERMANENT</validity>
                    <ltu>1</ltu>
                    <gracePeriod>-1</gracePeriod>
                    <graceCapacity>-1</graceCapacity>
                    <licenseModel>FLOATING</licenseModel>
                    <subscriptionRenewalPeriod>-1</subscriptionRenewalPer
iod>
                    <skuInfo>HP UFT</skuInfo>
                    <licenseKeyPurpose>For EPR Customers</licenseKeyPurpo
se>
                    <LicenseStatus>
```

```
                        <isBlacklisted>false</isBlacklisted>
                        <isActivated>false</isActivated>
                        <isExpired>false</isExpired>
                        <isTampered>false</isTampered>
                        <isIOValid>false</isIOValid>
                        <isLockValid>true</isLockValid>
                    </LicenseStatus>
                    <origin>
                        <system>APSC</system>
                        <user>bhimanagoud.parakanahalli@hpe.com</user>
                        <purpose>For EPR Customers</purpose>
                        <orderID>APSC</orderID>
                    </origin>
                    <feature>
                        <Id>10622</Id>
                        <version>1</version>
                    </feature>
                    <constraints>
                        <overrides/>
                        <dependsOn/>
                        <locationList/>
                        <lockList>
                            <lock>
                                <value>any</value>
                                <lockType>DEVICE_ID</lockType>
                            </lock>
                        </lockList>
                        <hostList/>
                        <supportedOSList/>
                    </constraints>
                    <attributeList/>
                    <machineDetails>
                        <hostname>457bdd6e30de</hostname>
                        <ipAddressList>
                            <ipAddresses>172.17.0.4</ipAddresses>
                            <ipAddresses>127.0.0.1</ipAddresses>
                            <ipAddresses>fe80:0:0:0:42:acff:fe11:4%eth0</
ipAddresses>
                        </ipAddressList>
                    </machineDetails>
                </license>
            </associatedLicenseList>
            <LiveCheckedoutCapacity>0</LiveCheckedoutCapacity>
            <LiveCheckedoutUnits>0.0</LiveCheckedoutUnits>
            <CommuterCheckedoutCapacity>0</CommuterCheckedoutCapacity>
            <CommuterCheckedoutUnits>0.0</CommuterCheckedoutUnits>
        </FeatureInfo>
    </LicensedFeatureList>
</ProductFeature>
```


401-UN AUTHORIZED	**In case if there is no IDM authentication token is passed,you will get 401 un authorized from the server**
400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue.**

Points to Note

- The response media can be either an xml or json based on the “Accept” header configuration in your query.

- The above API need to be authenticated using an IDM authentication token, please ensure you pass the “APAUTHTOKEN” for this query using the IDM authentication token.

Reservation Pool Users

Member Attributes

Member attributes should be passed when user call the API with *POST or DELETE methods* which contains *four attributes* data. Refer user guide for more details (**"Client User Management"**)

Refer the below table to understand member attributes for a given reservation pool:

ATTRIBUTE	DESCRIPTION
userName	**The Windows or Unix user name of a client user *DOMAIN/USERNAME*** **For example: ASIAPACIFIC/rpadmava**
ipAddress	**The IP address of the system from where the client accesses the AutoPass License Server.** **For example: 16.168.213.40**
host	**The host address of the client system** **For example: *RPADMAVA2***
clientID	**A unique value configured for each client based on the product’s support for this attribute.** **For example: RPADMAVA2 (any configured string from the client)**

Point to Note:

- The above four attributes can be encompassed as *<userNameList>, <ipAddressList>,<hostList>, </clientIDList>*, these lists are grouped under **“<members>”**
- The above attributes will be used to “Add/Delete/Get” member details from a reservation pool

Create Reservation Pool Members

This REST API will allow to add a list of members to already created reservation pool.
Adding members to APLS reservation pool can be done in two ways,

- By signing in *APLS UI as an administrator*. For more details how to create a pool and add users to a client pool please refer user guide (**"How to Manage Client User Access"**)
- By calling REST API by issuing the *HTTP POST* request to the *APLS endpoint/handler*. Please read the following details for more information about how to pass *POST request*.

Important information
Create a reservation pool through APLS UI in order for this rest service to successfully add the members. Please refer the user guide (**"Add a user pool"**).

Request

REQUEST

Method	URL
POST	https://<hostname>:<port>/autopass/wsservices/v9.3/reservation/pool/members

Request Examples:

- Use case: Add a list of domain users name to a pool.

```
<ReservationPool>
     <name>Development</name>
     <members>
            <userNameList>
                 <userName>ASIAPACIFIC/rpadmava</userName>
                 <userName>ASIAPACIFIC/ramana</userName>
                 <userName>ASIAPACIFIC/anantha</userName>
            </userNameList>
   </members>
</ReservationPool>
```

The above request contains the media type “application/xml” as an input.

The API is called with the above xml as an input with **HTTP POST** method to create the list of members for the pool.

You can also use combination of userName, ipAddress, host and clientID can be used to add to the reservation pool.

For example:

```
<ReservationPool>
     <name>Development</name>
     <members>
            <userNameList>
                 <userName>ASIAPACIFIC/rpadmava</userName>
                 <userName>ASIAPACIFIC/ramana</userName>
            </userNameList>
            <ipAddressList>
                 <ipAddress>16.168.213.40</ipAddress>
                 <ipAddress>16.168.213.41</ipAddress>
            </ipAddressList>
            <hostList>
                 <host>RPADMAVA2</host>
                 <host>RAMAN</host>
            </hostList>
            <clientIDList>
                 <clientID>RPADMAVA2</clientID>
                 <clientID>RAMANA2</clientID>
            </clientIDList>
   </members>
</ReservationPool>
```

To understand when we need to add a list of various other information, please check the “Client User Management” section in the APLS user guide.

Response

The Response by default will be in *XML* format. The status of this call can be identified through *status* field of

the response object which will contain SUCCESS / PARTIAL SUCCESS / FAIL.

If the API request to add members are successful, the members will be added to the specified pool in APLS *UI ->Reservation Management.*

STATUS	RESPONSE

201 Created	**Below is the response when the POST call to add members to a pool is successful.**
	
	`<ReservationPoolResult>`
	`  <name>Development</name>`
	`  <status>SUCCESS</status>`
	`  <summary>`
	`  <numberOfUsersUpdated>3/3</numberOfUsersUpdated>`
	`  </summary>`
	`</ReservationPoolResult>`

200 OK	**In case the second query is executed immediately after the first one, you will get the following response**
	
	`<ReservationPoolResult>`
	`  <name>Development</name>`
	`  <status>PARTIAL SUCCESS</status>`
	`  <summary>`
	`  <numberOfUsersUpdated>0/2</numberOfUsersUpdated>`
	`  <numberOfIPsUpdated>2/2</numberOfIPsUpdated>`
	`  <numberOfHostsUpdated>2/2</numberOfHostsUpdated>`
	`  <numberOfClientIDsUpdated>2/2</numberOfClientIDsUpdated>`
	`  </summary>`
	`  <errorList>`
	`  <errorDetail>`
	`  <errorCode>12002</errorCode>`
	`  <errorMessage>Member already exist</errorMessage>`
	`  <customMessage>User by (ASIAPACIFIC/rpadmava) already exists.</c`
	`ustomMessage>`
	`  </errorDetail>`
	`  <errorDetail>`
	`  <errorCode>12002</errorCode>`
	`  <errorMessage>Member already exist</errorMessage>`
	`  <customMessage>User by (ASIAPACIFIC/ramana) already exists.</cus`
	`tomMessage>`
	`  </errorDetail>`
	`  </errorList>`
	`</ReservationPoolResult>`

404 NOT FOUND	**In case the pool does not exist in APLS or URL is malformed**
	
	`<ReservationPoolResult>`
	`  <status>FAIL</status>`
	`  <errorList>`
	`  <errorDetail>`
	`  <errorCode>12001</errorCode>`
	`  <errorMessage>Pool Name doesn't exists.</errorMessage>`
	`  <customMessage>Pool by name({QA})doesn't exist at License Server`
	`</customMessage>`
	`  </errorDetail>`
	`  </errorList>`
	`</ReservationPoolResult>`

400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue.**

Points to note

- It is recommended to restrict the maximum number of members to be added to 100, for faster response.
- Currently the request input support the media type “application/xml” only. The response media can be either an xml or json based on the “Accept” header configuration in your query.
- Also only APLS admin users are authorized to execute this query, please ensure you pass the “Authorization” for this query using the admin users in APLS.
- Pool name is case sensitive.

Retrieve Pool Member List

This API will get the member values such as *User Name, IP Address, Host ID and Client ID* for a given Pool name.

Request

REQUEST

Method	URL
GET	**https://<hostname>:<port>/autopass/wsservices /v9.3/reservation/pool/members?poolName={pool_name}**

@param poolName**:**
Reservation pool as configured in Reservation Management -> Pool Management of AutoPass License Server. For more details to get the {pool_name}, please refer the user guide **("Pool Management Tab").**

Response

Response by default will be in the *XML format* which contains *HTTP* status code and member values for a given reservation pool name. Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE

200 OK	Below is the **success response** for the above query.
	Input value for {pool_name} in this API query is ‘Development’(*case sensitive*)
	`<ReservationPoolResult>`
	`  <name>Development</name>`
	`  <status>SUCCESS</status>`
	`  <members>`
	`  <userNameList>`
	`  <userName>ASIAPACIFIC/rpadmava</userName>`
	`  </userNameList>`
	`  <ipAddressList/>`
	`  <hostList/>`
	`  <clientIDList/>`
	`  </members>`
	`</ReservationPoolResult>`

200 OK	**In case there is no members configured for the reservation pool empty lists are returned as response** `<ReservationPoolResult>`

```
   <name>Development</name>
   <status>SUCCESS</status>
   <members>
      <userNameList/>
      <ipAddressList/>
      <hostList/>
      <clientIDList/>
   </members>
</ReservationPoolResult>
```

404 NOT FOUND — **Below response if the pool name is incorrect (Pool name does not exist)**

```
<ReservationPoolResult>
   <status>FAIL</status>
   <errorList>
      <errorDetail>
         <errorCode>12001</errorCode>
         <errorMessage>Pool Name doesn't exists.</errorMessage>
         <customMessage>Pool by name ({quality assurance}) doesn't exist at Licens
e Server</customMessage>
      </errorDetail>
   </errorList>
</ReservationPoolResult>
```

400 BAD REQUEST — **In case the input is malformed or not accepted, you will get 400 Bad Request from the server**

500 INTERNAL ERROR — **In case of APLS server is not reachable/down or any internal issue.**

Points to note

- The other format supported for response is JSON
- Pool name is case sensitive.

Delete Reservation Pool Members

This REST API will allow you to DELETE a list of member from a reservation pool.

Create a *HTTP DELETE* request to delete the attribute values from a given Pool Name. Authentication details must be also submitted as part of *HTTP* request header.

Deletion of members from APLS reservation pool can be done in two ways,

- By signing in to the *APLS UI*. For more details how to remove client user attributes from pool. Please refer user guide (**"How to Manage Client User Access"**)
- By calling the API by issuing the *HTTP DELETE* request to the *APLS* endpoint/handler.

By calling REST API by issuing the *HTTP POST* request to the *APLS endpoint/handler*. Please read the following details for more information about how to pass *POST request*.

Please read the following details for more information about how to pass DELETE request.

Important information

Ensure a reservation pool exists through APLS UI in order for this rest service to successfully delete the members. Please refer the user guide (**"Remove a client user's attribute from a pool"**).

The following is the End Point to be used with *HTTP DELETE* request. The request shall be in *XML* format.

Request

REQUEST

Method	URL
DELETE	https://<hostname>:<port>/autopass/wsservices/v9.3/reservation/pool/members

Sample Requests:

Use case remove list of domain users from a pool

```
<ReservationPool>
     <name>Development</name>
     <members>
            <userNameList>
                 <userName>ASIAPACIFIC/rpadmava</userName>
                 <userName>ASIAPACIFIC/ramana</userName>
                 <userName>ASIAPACIFIC/anantha</userName>
            </userNameList>
   </members>
</ReservationPool>
```

The above request contains the media type “application/xml” as an input.

The API is called with the above xml as an input with **HTTP DELETE** method to delete the list of members for the pool.

You can also use combination of username, ipAddress, host and clientID can be used to add to the reservation pool.

For example:

```
<ReservationPool>
     <name>Development</name>
     <members>
            <userNameList>
                 <userName>ASIAPACIFIC/rpadmava</userName>
                 <userName>ASIAPACIFIC/ramana</userName>
            </userNameList>
            <ipAddressList>
                 <ipAddress>16.168.213.40</ipAddress>
                 <ipAddress>16.168.213.41</ipAddress>
            </ipAddressList>
            <hostList>
                 <host>RPADMAVA2</host>
                 <host>RAMAN</host>
            </hostList>
            <clientIDList>
                 <clientID>RPADMAVA2</clientID>
                 <clientID>RAMANA2</clientID>
            </clientIDList>
   </members>
</ReservationPool>
```

Response

The Response by default will be in *XML* format which contain a *HTTP* status code and response XML.

Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE

200 OK	Below is the response when the DELETE call to remove members from a pool is successful
	`<ReservationPoolResult>`
	`<name>Development</name>`
	`<status>SUCCESS</status>`
	`<summary>`
	`<numberOfUsersUpdated>3/3</numberOfUsersUpdated>`
	`</summary>`
	`</ReservationPoolResult>`

200 OK	**In case the second query is executed immediately after the first one, you will get the following response**
	`<ReservationPoolResult>`
	`<name>Development</name>`
	`<status>PARTIAL SUCCESS</status>`
	`<summary>`
	`<numberOfUsersUpdated>0/2</numberOfUsersUpdated>`
	`<numberOfIPsUpdated>2/2</numberOfIPsUpdated>`
	`<numberOfHostsUpdated>2/2</numberOfHostsUpdated>`
	`<numberOfClientIDsUpdated>2/2</numberOfClientIDsUpdated>`
	`</summary>`

	<errorList> <errorDetail> <errorCode>12005</errorCode> <errorMessage>Member doent not exist</errorMessage> <customMessage>User by (ASIAPACIFIC/rpadmava) not exist in (Development)</customMessage> </errorDetail> <errorDetail> <errorCode>12005</errorCode> <errorMessage>Member doent not exist</errorMessage> <customMessage>User by (ASIAPACIFIC/ramana) not exist in (Development)</customMessage> </errorDetail> </errorList> </ReservationPoolResult></ReservationPoolResult>
404 NOT FOUND	**In case the pool does not exist in APLS** <ReservationPoolResult> <status>FAIL</status> <errorList> <errorDetail> <errorCode>12001</errorCode> <errorMessage>Pool Name doesn't exists.</errorMessage> <customMessage>Pool by name({QA})doesn't exist at License Server</customMessage> </errorDetail> </errorList> </ReservationPoolResult>
400 BAD REQUEST	**In case the input is malformed or not accepted, you will get 400 Bad Request from the server**
500 INTERNAL ERROR	**In case of APLS server is not reachable/down or any internal issue. Or a wrong URL is configured.**

Points to note

- It is recommended to restrict the total number of member values deleted to 100 for faster response.
- Currently the request input support the media type “application/xml” only. The response media can be either an xml or json based on the “Accept” header configuration in your query.
- Also only APLS admin users are authorized to execute this query, please ensure you pass the “Authorization” for this query using the admin users in APLS.
- Pool name is case sensitive.

License Usage

ATTRIBUTE	DESCRIPTION
productId	Products code of the product configured in the APLS For example: HP UFT
productVersion	Product's version For example: 12.52
pdfID	Product definition file ID. Internal purpose, please ignore this value.
pdfVersion	Product definition file version. Internal purpose, please ignore this value.
usageStartTimeUTCInSeconds	Start time of the duration for which usage is retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400
usageEndTimeUTCInSeconds	End time for the duration for which usage is retrieved.
featureId	Feature ID
featureVersion	Feature Version
featureType	Feature Type (FLOATING)
featureDescription	Feature Description
peakUsage	Peak usage during the duration
averageUsage	Average usage for the duration. Please *notice the web service calculates,* Average = Total capacity checked out for the duration / Number of check outs. In case of UI, the average calculated is per day and for the duration specified it is represented as, Average = ∑(Average(day(1)) …Average(day(n)) / number of days
totalCapacityCheckedout	Total capacity checked out for the duration
applicationName	Application name.
poolName	Reservation pool name
poolCreatedBy	Reservation pool created by

poolCreationTime	**Time the reservation pool created**
allotedCapacityInPool	**Reserved capacity of the pool**
isRestricted	**Is the members restricted to the pool's reserved capacity**

Points to note:

- The above attributes are encompassed such as PoolFeatureUsage, ApplicationUsage, and ProductFeaturePeakUsage in case of XML media type, based on the REST API

Retrieve Feature Usage

This REST API allows a developer to retrieve the usage for a Feature.

Request

REQUEST

Method	**URL**
GET	**https://<hostname>:<port>/autopass/wsservices/v9.3/usage/feature?featureId={feature_id}&featureVersion={feature_version}&startTime={startTimeInEpochSeconds}&endTime={startTimeInEpochSeconds}**

@**param** featureId:
Identifier of the feature for which usage is to be retrieved. Please refer to "License Usage" pane in APLS to identify the {feature_id} of the feature

@**param** featureVersion:
Version of the feature for which the usage is to be retrieved. Please refer the same "License Usage" pane in APLS to identify the {feature_version} of the feature

@**param** startTime:
Start time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

@**param** endTime:
End time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

Points to note:
By default, the response mediate type is of XML formation; however all GET APIs supports *JSON* format as well. To generate JSON the request's Accept header should have application/json.

Response

Response by default will be in the *XML format* which contains *HTTP* status code and usage for a given feature and duration.

Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE
200 OK	Below is the **success response** for the above query. Input value for {feature_id} is 10616, {feature_version} is 1, {startTimeInEpochSeconds} is 1475519400 and {startTimeInEpochSeconds} is 1475565690.

```
<ProductFeaturePeakUsage>
        <productId>HP</productId>
        <productVersion>12.52</productVersion>
        <pdfID>10027</pdfID>
        <pdfVersion>1.0</pdfVersion>
        <usageStartTimeUTCInSeconds>1475519400</usageStartTimeUTCInSeconds>
        <usageEndTimeUTCInSeconds>1475565690</usageEndTimeUTCInSeconds>
        <featureBasedUsage>
                <feature>
                        <featureId>10616</featureId>
                        <featureVersion>1</featureVersion>
                        <featureType>FLOATING</featureType>
                        <featureDescription>QuickTest Pro Java Add-in Concurrent
User</featureDescription>
                </feature>
                <capacityUsage>
                        <peakUsage>9</peakUsage>
                        <averageUsage>3</averageUsage>
                </capacityUsage>
                <checkoutsUsage>
                        <totalCapacityCheckedout>729</totalCapacityCheckedout>
                </checkoutsUsage>
        </featureBasedUsage>
         <status>SUCCESS</status>
</ProductFeaturePeakUsage
```

404 Not Found — **In case an invalid or unavailable feature's usage is queried**

```
<ProductFeaturePeakUsage>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12012</errorCode>
      <errorMessage>Product not found.</errorMessage>
      <customMessage>Product for feature 1061699:1 is not
configured in the License Server
      </customMessage>
    </errorDetail>
</ProductFeaturePeakUsage>
```

200 OK — **In case of valid feature but usage in not available for the duration**

```
<ProductFeaturePeakUsage>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12005</errorCode>
      <errorMessage>Resource does not exist</errorMessage>
      <customMessage>Usage for resource 10594:1 is not found.</customMess
age>
   </errorDetail>
</ProductFeaturePeakUsage>
```

Points to note

- If client time out is set to 60 seconds and there are more than 1500 license transactions during the specified period, the response may timeout. Please ensure you increase the connection and read timeout in this case

Retrieve Reservation Pool Usage

This REST API allows a developer to retrieve the usage for a reservation pool.

Request

REQUEST

Method	URL
GET	**https://<hostname>:<port>/autopass/wsservices/v9.3/usage/pool?poolName ={pool_name}&startTime={startTimeInEpochSeconds}&endTime={startTimeInEpochSeconds}**

@**param** poolName:
Reservation pool as configured in Reservation Management -> Pool Management of AutoPass License Server. For more details to get the {pool_name} information, please refer the user guide **("Pool Management Tab").** *Pool name is case sensitive*. To retrieve usage for the common pool, specify "*Common Pool*" (case insensitive)

@**param** startTime:
Start time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

@**param** endTime:
End time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

Points to note:
By default, the response mediate type is of XML formation; however all GET APIs supports *JSON* format as well. To generate JSON the request's Accept header should have application/json.

Response

Response by default will be in the *XML format* which contains *HTTP* status code and usage for a given feature and duration.

Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE
200 OK	Below is the **success response** for the above query. Input value for {pool_name} is Development, {startTimeInEpochSeconds} is 1475519400 and {startTimeInEpochSeconds} is 1475565690.

```
<PoolFeatureUsage>
        <poolName>Development</poolName>
        <usageStartTimeUTCInSeconds>1473967740</usageStartTimeUTCInSeconds>
        <usageEndTimeUTCInSeconds>1475561488</usageEndTimeUTCInSeconds>
        <featureUsageDetails>
           <poolUsage>
             <poolFeatureDetails>
                <featureId>10606</featureId>
                <featureVersion>1</featureVersion>
```

```
                <featureType>FLOATING</featureType>
                <featureDescription>QuickTest Professional Core Concurrent User</
featureDescription>
                <allotedCapacityInPool>500</allotedCapacityInPool>
                <isRestricted>false</isRestricted>
            </poolFeatureDetails>
                <capacityUsage>
                  <peakUsage>243</peakUsage>
                  <averageUsage>2</averageUsage>
                </capacityUsage>
                <checkoutsUsage>
                    <totalCapacityCheckedout>2636</totalCapacityCheckedout>
                </checkoutsUsage>
          </poolUsage>
        </featureUsageDetails>
        <status>SUCCESS</status>
</PoolFeatureUsage>
```

404 Not Found — **In case an invalid or unavailable pool usage is queried**

```
<PoolFeatureUsage>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12005</errorCode>
      <errorMessage>Resource does not exist</errorMessage>
      <customMessage>Resource Development1 does not exist.</customMessag>
   </errorDetail>
</PoolFeatureUsage>
```

200 OK — **In case of valid pool name but usage in not available for the duration**

```
<PoolFeatureUsage>
   <poolName>Development</poolName>
   <poolCreatedBy>admin</poolCreatedBy>
   <poolCreationTime>1475668559</poolCreationTime>
   <usageStartTimeUTCInSeconds>1475692200</usageStartTimeUTCInSeconds>
   <usageEndTimeUTCInSeconds>1475778599</usageEndTimeUTCInSeconds>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12005</errorCode>
      <errorMessage>Resource does not exist</errorMessage>
      <customMessage>Usage for resource Development is not found.</custom
Message>
   </errorDetail>
</PoolFeatureUsage>
```

Points to note:

- Single pool usage can be retrieved through one request.
- Pool name other than “Common Pool” is case sensitive.
- If client time out is set to 60 seconds and there are more than 1500 license transactions during the specified period, the response may timeout. Please ensure you increase the connection and read timeout in this case

Retrieve Application Usage

This REST API allows a developer to retrieve usage against the feature which have been checked out for the given application name from the client.

Request

REQUEST

Method	URL
GET	https://<hostname>:<port>/autopass/wsservices/v9.3/usage/application?applicationName={application_name}&startTime={startTimeInEpochSeconds}&endTime={startTimeInEpochSeconds}

@param applicationName:
Application name is the value that product may pass during each check out transaction. .
For more details to get the {application_name} please refer the user guide **("Feature Report Page (License Usage Pane")** table. Application name is case insensitive.

@param startTime:
Start time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

@param endTime:
End time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400

Points to note:
By default, the response mediate type is of XML formation; however all GET APIs supports *JSON* format as well. To generate JSON the request's Accept header should have application/json.

Response

Response by default will be in the *XML format* which contains *HTTP* status code and usage for a given feature and duration.

Also the status of this call can be identified through status field of the response object which will contain SUCCESS / FAIL.

STATUS	RESPONSE
200 OK	Below is the **success response** for the above query. Input value for {application_name} is LeanFT, {startTimeInEpochSeconds} is 1475605800 and {startTimeInEpochSeconds} is 1475692199.

```
<ApplicationUsage>
   <applicationName>LeanFT</applicationName>
   <usageStartTimeUTCInSeconds>1475605800</usageStartTimeUTCInSeconds>
   <usageEndTimeUTCInSeconds>1475692199</usageEndTimeUTCInSeconds>
   <productDetails>
      <productId>HP UFT</productId>
      <productVersion>12.52</productVersion>
      <pdfID>10027</pdfID>
      <pdfVersion>1.0</pdfVersion>
      <featureUsage>
         <feature>
```

```
            <featureId>10594</featureId>
            <featureVersion>1</featureVersion>
            <featureType>FLOATING</featureType>
            <featureDescription>HP Unified Functional Testing Concurrent User</fea
tureDescription>
         </feature>
         <capacityUsage>
            <peakUsage>5</peakUsage>
            <averageUsage>1</averageUsage>
         </capacityUsage>
         <checkoutsUsage>
            <totalCapacityCheckedout>10</totalCapacityCheckedout>
         </checkoutsUsage>
      </featureUsage>
   </productDetails>
   <status>SUCCESS</status>
</ApplicationUsage>
```

404 Not Found — **In case an invalid or unavailable application usage is queried**

```
<ApplicationUsage>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12005</errorCode>
      <errorMessage>Resource does not exist</errorMessage>
      <customMessage>Usage for resource SomeVale is not found.</customMessage>
      </errorDetail>
</ApplicationUsage>
```

200 OK — **In case of valid application name but usage in not available for the duration**

```
<ApplicationUsage>
   <status>FAIL</status>
   <errorDetail>
      <errorCode>12005</errorCode>
      <errorMessage>Resource does not exist</errorMessage>
      <customMessage>Usage for resource LeanFT is not found.</customMessage>
      </errorDetail>
</ApplicationUsage>
```

Points to note:

- Single application usage can be retrieved through one request.
- Application name is case insensitive.
- If client time out is set to 60 seconds and there are more than 1500 license transactions during the specified period, the response may timeout. Please ensure you increase the connection and read timeout in this case

Feature Usage Data

ATTRIBUTE	DESCRIPTION
featureId	**Feature ID**
featureVersion	**Feature Version**
usageStartTimeInUTCSec	**Start time of the duration for which usage is retrieved.** **Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400.**
usageEndTimeInUTCSec	**End time for the duration for which usage is retrieved.**
reportType	**(1) Report Type '0' retrieves the license check out report and details for a specific feature from 'In Use' page. (2) Report Type '1' retrieves the license checked in report and details for a specific feature from 'History' page.**
poolName	**Reservation pool name. The default is 'ALL'**
startAt	**The startAt (Also called Page Number) starts with '1'. When a query (GET) would return a large chunk of data, the result is broken into 'pages'. 'startAt'– the row/record number of the first item returned (in the transactionDetailsList attribute).**
count	**The actual number of resources returned. By default it return 200 records.**

Points to note:

- The above attributes are encompassed such as Raw and Summary Transaction details in case of XML media type, based on the REST API

Retrieve Feature Usage Raw Transaction

This REST API allows a developer to retrieve the license checked out report and details for a specific feature. When a license is checked out and checked in for a particular feature the usage data is also reflected in the 'In Use' and' History' page respectively. To retrieve the details of a specific feature license then call this API to get the Feature Report details. The Feature Report details displays specific check out information about a feature license including:

(1) The checkout start and expiration date.
(2) The type of check out distribution: Live or Commuter.
(3) The capacity checked out.
(4) Check out and check in details for the license.
(5) The pools to which users who have licenses checked out belong. User can get the Usage Raw information based on required pool.

Request

REQUEST

Method	URL
GET	**https://<hostname>:<port>/autopass/services/v10.3/feature/usage/raw?featureId={feature_id}&featureVersion={feature_version}&usageStartTimeInUTCSec={startTimeInEpochSeconds}&usageEndTimeInUTCSec={endTimeInEpochSeconds}&reportType={ReportType}&poolName={Pool Name}&startAt={pageNumber}&count={Count of records to print}**

@**param** featureId:
Identifier of the feature for which usage is to be retrieved. Please refer to “License Usage” pane in APLS to identify the { featureId} of the feature.

@**param** featureVersion:
Version of the feature for which the usage is to be retrieved. Please refer the same “License Usage” pane in APLS to identify the {featureVersion} of the feature.

@**param** usageStartTimeInUTCSec:
Start time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400.

@**param** usageEndTimeInUTCSec:
End time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400.

@**param** reportType: **Default is 1**

(1) Report Type '0' Retrieves the license checked out report and details for a specific feature in 'In Use' page.
(2) Report Type '1' Retrieves the license checked in report and details for a specific feature in 'History' page.

@**param** poolName: **Default is ‘ALL’**
Reservation pool as configured in Reservation Management -> Pool Management of AutoPass License Server. For more details to get the {poolName} information, please refer the user guide ('Pool Management Tab').Pool name is case sensitive. To retrieve usage for the common pool, specify “Common Pool” (case insensitive).

@**param** startAt: **Default is 1**
The page number starts from 1 so when a query (GET) would return a large chunk of data, the result is broken into 'pages'. 'startAt'– also called page number of row/record number of the first item returned (in the transactionDetailsList attribute).

@**param** count: **Default is 200**
The actual number of resources returned (in the transactionDetailsList attribute). Default value is 200 per page.

Points to note:
By default, the response media type is of XML formation; however all GET APIs supports JSON format as well. To generate JSON the request’s Accept header should have application/json.

Response

Response by default will be in the *XML format* which contains *HTTP* status code and raw usage for a given feature and duration.

STATUS	RESPONSE
200 OK	Below is the **success response** for the above query. Input value for {featureId} is 10594, {featureVersion} is 1, {usageStartTimeInUTCSec} is 1505986228 ,{usageEndTimeInUTCSec} is 1505986228 {reportType} is 1 and {poolName} is ALL

```
<featureUsageDeatilResponse>
    <featureId>10594</featureId>
    <featureVersion>1</featureVersion>
    <featureDescription>HP Unified Functional User</featureDescription>
    <usageStartTimeInUTCSec>1505986228</usageStartTimeInUTCSec>
    <usageEndTimeInUTCSec>1505986228</usageEndTimeInUTCSec>
    <reportDate>21-Sep-2017 15:01:35</reportDate>
    <poolName>ALL</poolName>
    <reportType>1</reportType>
    <filters>
        <totalRecords>2</totalRecords>
        <count>2</count>
        <startAt>1</startAt>
    </filters>
    <transactionDetailsList>
        <licenseToken>
            <startDate>Thu Sep 21 15:00:09 IST 2017</startDate>
            <expiryDate>Sat Sep 23 15:00:09 IST 2017</expiryDate>
            <checkoutType>Commuter</checkoutType>
            <checkedOutCapacity>1.0</checkedOutCapacity>
            <hostIP>16.183.89.228</hostIP>
            <userName>IWFVM00428/Administrator</userName>
            <hostAddress>IWFVM00428.</hostAddress>
            <productName>HP UFT</productName>
            <productVersion>12.52</productVersion>
            <checkedOutBy>IWFVM00428.</checkedOutBy>
            <checkedInBy>IWFVM00428.</checkedInBy>
            <checkedInTime>Thu Sep 21 15:00:10 IST 2017</checkedInTime>
            <pool>Common Pool</pool>
        </licenseToken>
        <licenseToken>
            <startDate>Thu Sep 21 14:25:02 IST 2017</startDate>
            <expiryDate>Sat Sep 23 14:25:02 IST 2017</expiryDate>
            <checkoutType>Commuter</checkoutType>
            <checkedOutCapacity>1.0</checkedOutCapacity>
            <hostIP>16.183.89.228</hostIP>
            <userName>IWFVM00428/Administrator</userName>
            <hostAddress>IWFVM00428.</hostAddress>
            <productName>HP UFT</productName>
            <productVersion>12.52</productVersion>
            <checkedOutBy>IWFVM00428.</checkedOutBy>
            <checkedInBy>IWFVM00428.</checkedInBy>
            <checkedInTime>Thu Sep 21 14:25:42 IST 2017</checkedInTime>
            <pool>Common Pool</pool>
        </licenseToken>
    </transactionDetailsList>
</featureUsageDeatilResponse>
```

406	**In case an empty or unavailable feature's is queried**
Not Acceptable	

```
<featureUsageDeatilResponse>
    <errorDetail>
        <errorCode>14006</errorCode>
        <errorMessage>Invalid input</errorMessage>
```

```
        <customMessage>Feature ID cannot be null or empty.</customMessage
>
    </errorDetail>
</featureUsageDeatilResponse>
```

200 OK — **In case of valid feature but raw usage in not available for the duration**

```
<featureUsageDeatilResponse>
    <featureId>10606</featureId>
    <featureVersion>19</featureVersion>
    <usageStartTimeInUTCSec>1505986228</usageStartTimeInUTCSec>
    <usageEndTimeInUTCSec>1505986228</usageEndTimeInUTCSec>
    <reportDate>21-Sep-2017 15:09:28</reportDate>
    <poolName>ALL</poolName>
    <reportType>1</reportType>
    <filters>
        <totalRecords>0</totalRecords>
        <count>0</count>
    </filters>
    <transactionDetailsList/>
    <errorDetail>
        <errorCode>12006</errorCode>
        <errorMessage>No Records Found in License Server</errorMessage>
        <customMessage>Usage data is empty,No Records Found in Server</cu
stomMessage>
    </errorDetail>
</featureUsageDeatilResponse>
```

404 — **In case of not matching request uri**

```
The server has not found anything matching the Request-URI
```

405

```
The method specified in the Request-Line is not allowed for the resource i
dentified by the Request-URI.
```

Retrieve Feature Usage Summary Transaction

This REST API allows a developer to retrieve the summary of check out and check in history details as displayed in APLS UI. You can get this data based on the Group/Pool Name. It retrieves the following information.

(1) The name of the pool
(2) Pool wise usage maximum consumed or maximum checkouts at any point of time in the given date range.
(3) Average - Sum of all the checkouts divided by number of checkouts in the given date range.
(4) The number of users. The feature wise checked in and checked out history details in numbers at any given point of time i.e. (1) Maximum (2) Average

Request

REQUEST

Method	URL
GET	**https://<hostname>:<port>/autopass/services/v10.3/feature/usage/summary?featureId={feature_id}&featureVersion={feature_version}&usageStartTimeInUTCSec={startTimeInEpochSeconds}&usageEndTimeInUTCSec={endTimeInEpochSeconds&poolName={Pool Name}**

@**param** featureId:
Identifier of the feature for which usage is to be retrieved. Please refer to “License Usage” pane in APLS to identify the {featureId} of the feature.

@**param** featureVersion:
Version of the feature for which the usage is to be retrieved. Please refer the same “License Usage” pane in APLS to identify the {featureVersion} of the feature.

@**param** usageStartTimeInUTCSec:
Start time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400.

@**param** usageEndTimeInUTCSec:
End time of the duration for which the usage is to be retrieved. Time is represented as seconds using the standard base time known as "the epoch", for e.g. January 1, 2016, 00:00:00 GMT is represented as 1451606400.

@**param** poolName: Default is ‘ALL’
Reservation pool as configured in Reservation Management -> Pool Management of AutoPass License Server. For more details to get the {poolName} information, please refer the user guide ('Pool Management Tab').Pool name is case sensitive. To retrieve usage for the common pool, specify “Common Pool” (case insensitive).

Points to note:
By default, the response media type is of XML formation; however, all GET APIs supports JSON format as well. To generate JSON the request’s Accept header should have application/json.

Response
Response by default will be in the XML format which contains HTTP status code and summary information for a given feature and duration.

STATUS	RESPONSE

200 OK	Below is the **success response** for the above query.
	Input value for {featureId} is 10594, {featureVersion} is 1, {usageStartTimeInUTCSec} is 1505986228 ,{usageEndTimeInUTCSec} is 1505986228 ,and {poolName} is ALL
	`<featureUsageSummaryResponse>`
	`<featureId>10594</featureId>`
	`<featureVersion>1</featureVersion>`
	`<featureDescription>HP Unified Functional User</featureDescription>`
	`<usageStartTimeInUTCSec>1505986228</usageStartTimeInUTCSec>`
	`<usageEndTimeInUTCSec>1505986228</usageEndTimeInUTCSec>`
	`<reportDate>21-Sep-2017 15:20:40</reportDate>`
	`<poolName>all</poolName>`
	`<maximumCapacity>1.0</maximumCapacity>`
	`<averageCapacity>1.0</averageCapacity>`
	`<summarytransactionDetailsList>`
	`<transactionDetails>`
	`<usagedate>21-September-2017</usagedate>`
	`<maximumCapacity>1.0</maximumCapacity>`
	`<averageCapacity>1.0</averageCapacity>`
	`<numOfUniqueUsers>1</numOfUniqueUsers>`
	`</transactionDetails>`
	`</summarytransactionDetailsList>`
	`</featureUsageSummaryResponse>`

406
Not Acceptable
	In case an empty or unavailable feature’s is queried
	`<featureUsageSummaryResponse>`
	`<errorDetail>`
	`<errorCode>14006</errorCode>`
	`<errorMessage>Invalid input</errorMessage>`
	`<customMessage>Feature ID cannot be null or empty.</customMessage>`
	`</errorDetail>`

```
</featureUsageSummaryResponse>
```

200 OK — **In case of valid feature but raw usage in not available for the duration**

```
<featureUsageSummaryResponse>
    <featureId>10606</featureId>
    <featureVersion>19</featureVersion>
    <usageStartTimeInUTCSec>1505986228</usageStartTimeInUTCSec>
    <usageEndTimeInUTCSec>1505986228</usageEndTimeInUTCSec>
    <poolName>ALL</poolName>
    <summarytransactionDetailsList/>
    <errorDetail>
        <errorCode>12006</errorCode>
        <errorMessage>No Records Found in License Server</errorMessage>
        <customMessage>Summary data is empty,No Records Found in Server</
customMessage>
    </errorDetail>
</featureUsageSummaryResponse>
```

404 — **In case of not matching request uri**

```
The server has not found anything matching the Request-URI
```

405

```
The method specified in the Request-Line is not allowed for the resource i
dentified by the Request-URI.
```

GET Summary Product/License Usage

Report Summary usage

This API will retreive the usage reported by product, the installed licenses and license capacities for license features and product features**.**

- The product should be available in APLS before a usage is retrieved any of the product features or license features.
- This API supports both DB and IDM authentication (effective 10.6 version).

End point

Method	URL
POST	**https://<HOSTNAME >:<PORT>>/autopass/services/v10.5/feature/summary**

Request

Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="LicenseSummaryRequest">
    <xs:sequence>
      <xs:element name="pdfIdentifier" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Sample Request

```
<LicenseSummaryRequest>
        <pdfIdentifier>10027_1.0_HP UFT_12.50</pdfIdentifier>
</LicenseSummaryRequest>
```

Response

Schema

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="./Common.xsd" />
  <xs:include schemaLocation="./LicenseFeatureDetails.xsd" />
  <xs:complexType name="licenseSummaryResponse">
    <xs:sequence>
      <xs:element name="licenseFeatureDetails" type="LicenseFeatureDetails" maxOccurs="500"/>
      <xs:element name="errorDetail" type="ErrorDetail" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

STATUS	RESPONSE
200 OK	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <licenseSummaryResponse> <licenseFeatureDetails> <featureId>23266</featureId> <featureVersion>1</featureVersion> <featureDescription>HCM Trial Edition</featureDescription> <availableCap>1.0</availableCap> <issuedCap>0.0</issuedCap> <totalCap>1.0</totalCap> <commuteCap>0.0</commuteCap> <commuter>1</commuter> <adminCommuterDays>0</adminCommuterDays> <adminCommuterCapacity>0</adminCommuterCapacity> <minCapacity>1.0</minCapacity> <totalInstalledLicenseCap>1.0</totalInstalledLicenseCap> <licenseFeatureType>LICENSE_FEATURE</licenseFeatureType> <id>10</id> </licenseFeatureDetails> </licenseSummaryResponse>

JAVA Sample Code

Fetch License (“HTTP GET”) from APLS

```
StringBuffer urlString = new
StringBuffer("https://<APLM_IP_OR_HOST>:<PORT>/autopass/services/v1/license")
                            .append("?")
                            .append("productUniqueId=")
                            .append("<Unique_Product_Id>")
                            .append("lastServerTimeStamp=")
                            .append("<lastServerTimeStamp_In_seconds>");

URL url = new URL strBuffer.toString());

HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("GET");
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");

OutputStream os = conn.getOutputStream();
os.write(input.toString().getBytes());
os.flush();

If not created
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    throw new RuntimeException("Failed : HTTP error code : "
                            + conn.getResponseCode());
}

BufferedReader br = new BufferedReader(new InputStreamReader(
                            (conn.getInputStream())));

String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
              System.out.println(output);
}

conn.disconnect();
```

Remove License (“HTTP DELETE”) from APLS

```
URL url = new URL("https://<APLM_IP_OR_HOST>:<PORT>/autopass/services/v1/licenses");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("DELETE");//DELETE to remove licenses from the APLMS
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");

//Input xml
StringBuffer input = new StringBuffer("<RemoveLicenseRequest>")
                            .append("<RemoveLicenseInfo>")
                            .append("<featureID>").append(Featureid).append("</featureID>")
```

```
                            .append ("<featureVersion>1</featureVersion>     "))
                            .append ("</RemoveLicenseInfo>"))
                            .append ("</RemoveLicenseRequest>");

conn.setRequestProperty("APAUTHTOKEN", getIDMauthToken());

OutputStream os = conn.getOutputStream();
os.write(input.toString().getBytes());
os.flush();

If not created
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    throw new RuntimeException("Failed : HTTP error code : "
                            + conn.getResponseCode());
}

BufferedReader br = new BufferedReader(new InputStreamReader(
                            (conn.getInputStream())));

String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
              System.out.println(output);
}

conn.disconnect();
```

Add Licenses (“HTTP POST”) to APLS

```
private static String TARGET_URL =https://<APLM_IP_OR_HOST>:<PORT>/autopass/services/v1/";
String upload = "C:/file.txt";
ClientConfig clientConfig = new ClientConfig();
Client client = ClientBuilder.newBuilder().register(MultiPartFeature.class).build();
client.register(feature);
clientConfig.register(MultiPartFeature.class);
WebTarget target = client.target(TARGET_URL).path("licenses");
FormDataMultiPart multiPart = new FormDataMultiPart();
FileDataBodyPart fileDataBodyPart = new FileDataBodyPart("file", new File(upload));
 multiPart.bodyPart(fileDataBodyPart);
 System.out.println ("Uploading license file...");
 Response response = null;
   try {
     response=target.request().header("APAUTHTOKEN",getIDMauthToken()).post(Entity.entity(multiP
art, multiPart.getMediaType()), Response.class);
    } catch (Exception ex) {
                System.out.println(ex);
    }
     if (response != null) {
                System.out.println("Upload response: " + response.getStatus() + " " +
response.getStatusInfo() + " " + response);
       }
```

Get the list of licenseID's from AutoPassJ Core and use the licenseID in removeAPI call to remove the license(s) from APLS based on LicenseID

```
Properties autopassjprop=new Properties();
autopassjprop.setProperty(AutopassJPropertyKeys.DATA_DIR,"C://data");//data dirctory
autopassjprop.setProperty(AutopassJPropertyKeys.LIC_FILE, "C://data/LicFile.txt");//lic file
autopassjprop.setProperty(AutopassJPropertyKeys.PDF_PATH, "C:/10027_1.0_HP UFT_12.52.pd");//pd
autopassjprop.setProperty(AutopassJPropertyKeys.ENABLE_CRYPTO_TYPE, "0");
autopassjprop.setProperty(AutopassJPropertyKeys.ALLOW_APSC_GENERATED_KEY, "N");
try
{
AutopassJ autopassj=new AutopassJ(autopassjprop);
Lock lock=new Lock();
List<License> licenseList=autopassj.reportLicenseSet (lock, false);
for(License license: licenseList) {
System.out.println("License ID" + license.getLicenseID());
}
catch(Exception ex)
{System.out.println(ex);}
```

Get Lock ("HTTP POST") from APLS

```
URL url = new URL("https://<APLM_IP_OR_HOST>:<PORT>/autopass/services/v1/lock");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("POST");//DELETE to remove licenses from the APLMS
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");

//Input xml
StringBuffer input = new StringBuffer("<LmsLockRequest>")
                          .append("<productUniqueId>")
                          .append("<productUniqueId>")
                          .append("</productUniqueId>")
                          .append ("</LmsLockRequest>");

OutputStream os = conn.getOutputStream();
os.write(input.toString().getBytes());
os.flush();

If not created
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    throw new RuntimeException("Failed : HTTP error code : "
                          + conn.getResponseCode());
}

BufferedReader br = new BufferedReader(new InputStreamReader(
                          (conn.getInputStream())));

String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
             System.out.println(output);
}
```

```
conn.disconnect();
```

Get Feature Details (“HTTP GET”) from APLS

```
StringBuffer urlString = new
StringBuffer("https://<APLM_IP_OR_HOST>:<PORT>/autopass/services/v1/product/features")
                            .append("?")
                            .append("productUniqueId=")
                            .append("HPEOBR_10.10");

URL url = new URL strBuffer.toString());

HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("GET");//DELETE to remove licenses from the APLMS
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");
conn.setRequestProperty("APAUTHTOKEN", getIDMauthToken());


OutputStream os = conn.getOutputStream();
os.write(input.toString().getBytes());
os.flush();

If not created
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
                    throw new RuntimeException("Failed : HTTP error code : "
                            + conn.getResponseCode());
}

BufferedReader br = new BufferedReader(new InputStreamReader(
                            (conn.getInputStream())));

String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
              System.out.println(output);
}

conn.disconnect();
```

Sample code to “GET” feature usage

```
URL url = new
URL("https://localhost:5814/autopass/wsservices/v9.3/usage/feature?featureId=10616&featureVersio
n=1&startTime=1475605800&endTime=1475692199");

HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/xml");

//Authorization Header
String aplsUserName = "aplsUser";
String aplsPassword = "password";
BASE64Encoder enc = new sun.misc.BASE64Encoder();
```

```
String userpassword = new
StringBuffer(aplsUserName).append(":").append(aplsPassword).toString();
String encodedAuthorization = enc.encode(userpassword.getBytes());
StringBuffer encodeValue = new StringBuffer("Basic ").append(encodedAuthorization);
conn.setRequestProperty("Authorization", encodeValue.toString());


//Connect to Server
conn.connect();

if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
       throw new RuntimeException("Failed : HTTP error code : "
              + conn.getResponseCode());
}

//Get response
BufferedReader br = new BufferedReader(new InputStreamReader(
       (conn.getInputStream())));

//Print response
String output;
System.out.println("Output from Server .... \n");
while ((output = br.readLine()) != null) {
   System.out.println(output);
}

//Disconnect Server
onn.disconnect();
```

Sample code to "POST/DELETE" (add/remove) members to/from Reservation Pool.

```
URL url = new URL("https://localhost:5814/autopass/wsservices/v9.3/reservation/pool/members");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("POST");//DELETE to remove members from the reservation pool
conn.setRequestProperty("Content-Type", "application/xml");
conn.setRequestProperty("Accept", "application/xml");

//Input xml
StringBuffer input = new StringBuffer("<ReservationPool>")
                          .append("<name>Development</name>")
                          .append("<members>")
                          .append("<userNameList>")
                          .append("<userName>ASIAPACIFIC/rpadmava</userName>")
                          .append("<userName>ASIAPACIFIC/ramana</userName>")
                          .append("</userNameList>")
                          .append("</members>")
                          .append("</ReservationPool>");

//Admin Authorization Header
String aplsUserName = "admin";
String aplsPassword = "password";
BASE64Encoder enc = new sun.misc.BASE64Encoder();
String userpassword = new
StringBuffer(aplsUserName).append(":").append(aplsPassword).toString();
String encodedAuthorization = enc.encode(userpassword.getBytes());
StringBuffer encodeValue = new StringBuffer("Basic ").append(encodedAuthorization);
conn.setRequestProperty("Authorization", encodeValue.toString());

OutputStream os = conn.getOutputStream();
os.write(input.toString().getBytes());
os.flush();

If not created
if (conn.getResponseCode() != HttpURLConnection.HTTP_CREATED || conn.getResponseCode() !=
HttpURLConnection.HTTP_OK) {
                    throw new RuntimeException("Failed : HTTP error code : "
                          + conn.getResponseCode());
}

BufferedReader br = new BufferedReader(new InputStreamReader(
                          (conn.getInputStream())));

String output;
System.out.println("Output from Server .... \n");
while ((output = br.readLine()) != null) {
             System.out.println(output);
}

conn.disconnect();
```

Sample code to “GET” feature usage Raw Transaction Data

```
URL url = new
URL("https://localhost:5814/autopass/services/v10.3/feature/usage/raw?featureId=10594&featureVer
sion=1&startTime=1505986228&endTime=1505986228&reportType=1&poolName=all");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/xml");

//Authorization Header
String aplsUserName = "admin";
String aplsPassword = "password";
BASE64Encoder enc = new sun.misc.BASE64Encoder();
String userpassword = new
StringBuffer(aplsUserName).append(":").append(aplsPassword).toString();
String encodedAuthorization = enc.encode(userpassword.getBytes());
StringBuffer encodeValue = new StringBuffer("Basic ").append(encodedAuthorization);
conn.setRequestProperty("Authorization", encodeValue.toString());

//Connect to Server
conn.connect()
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
       throw new RuntimeException("Failed : HTTP error code : "
              + conn.getResponseCode());
}

//Get response
BufferedReader br = new BufferedReader(new InputStreamReader(
       (conn.getInputStream())));

//Print response
String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
   System.out.println(output);
}

//Disconnect Server
onn.disconnect();
```

Sample code to “GET” feature usage summary Transaction Data

```
URL url = new
URL("https://localhost:5814/autopass/services/v10.3/feature/usage/summary?featureId=10594&featur
eVersion=1&startTime=1505986228&endTime=1505986228&poolName=all");
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setDoOutput(true);
conn.setRequestMethod("GET");
conn.setRequestProperty("Accept", "application/xml");

//Authorization Header
String aplsUserName = "admin";
String aplsPassword = "password";
BASE64Encoder enc = new sun.misc.BASE64Encoder();
String userpassword = new
StringBuffer(aplsUserName).append(":").append(aplsPassword).toString();
```

```
String encodedAuthorization = enc.encode(userpassword.getBytes());
StringBuffer encodeValue = new StringBuffer("Basic ").append(encodedAuthorization);
conn.setRequestProperty("Authorization", encodeValue.toString());

//Connect to Server
conn.connect()
if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
       throw new RuntimeException("Failed : HTTP error code : "
              + conn.getResponseCode());
}

//Get response
BufferedReader br = new BufferedReader(new InputStreamReader(
       (conn.getInputStream())));

//Print response
String output;
System.out.println ("Output from Server .... \n");
while ((output = br.readLine()) != null) {
   System.out.println(output);
}

//Disconnect Server
onn.disconnect();
```

Points to note:

1. Use any Base64 encode options available to encode your username and password.
2. For “GET” APIs, “application/json” option is also available for header
3. HTTPS connection requires client side certificate verification. It is recommended to have a valid certificate at server and implement the certificate validation at client side. Refer “Using SSL Authentication in Java Clients” for Java implementation
4. Search for “Developing RESTful Web Service Clients”, to implement RESTful clients
5. Important point to note, if client time out is set to 60 seconds and there are more than 1500 license transactions during the specified period, the response may timeout. Please ensure you increase the connection and read timeout in this case
6. The schema for the requests and responses are available at the APLS installer path <INSTALLED_PATH>\HP AutoPass License Server\HP AutoPass License Server\webapps\autopass\sdk\xsd

Error Codes

ERROR CODE	ERROR INFO	CUSTOM ERROR INFO
15010	Feature or licenseID doesn't match.	licenseID ({value}) specified is not associated with given Feature {value}.
15005	Invalid License Usage.	{0} Invalid License Usage: License(s) has been generated from APSC (AutoPass Service Center).
15006	License(s) not added as it cannot run in FIPS Mode.	{0} License(s) not added as it cannot run in FIPS Mode. FIPS mode have to be disabled to run these license(s).
15001	Invalid License File.	Invalid License File. The file contains invalid licenses.
15007	No License found.	License(s) not found in APLM server.
15008	License list is not defined.	License list is not defined.
14008	Product Definition is not found for the product in the APLM server.	Product Definition is not found for feature 206218:1 in the APLM server
14001	The format of APLM server timestamp must be long.	The format of APLM server timestamp must be long.
14009	Product unique identifier cannot be null or empty.	Product unique identifier cannot be null or empty.
15011	Null or empty productUniqueId or (productId and productVersion) or (featureId or featureVersion)	Null or empty productUniqueId or (productId and productVersion) or (featureId or featureVersion)
12001	Pool Name does not exists	Pool by name ({pool_name}) doesn't exist at License Server
12002	Member already exist	• User by (x) already exists • IP by (x) already exists • Host by (x) already exists • Client ID by (x) already exists
12003	Member list is not defined	{userNameList} {ipAddressList} {hostList} {clientIDList} attributes are not defined
12004	IP address is not valid	Please enter valid IP address

12005	**Member doesn't not exist**	• User by **{value)** doesn't exist in **(pool1)** • IP by **{value)** doesn't exist in **(pool1)** • Host by **{value)** doesn't exist in **(pool1)** • ClientID by **{value)** doesn't exist in **({pool_name})**
12006	**Failed to add member**	• Empty message string is passed by User. Field value should not be null or empty • Empty message string is passed by IP. Field value should not be null or empty • Empty message string is passed by Host. Field value should not be null or empty • Empty message string is passed by Client Id. Field value should not be null or empty
12010	**Invalid input**	• Invalid value {value} for query parameter {@paramname}
12011	**Authorization Failed**	• User - {value}, does not have permission to perform this action
12012	**Product not found**	• Product {product_code} with version {product_version} is not configured in the License Server

Acronyms

APLS AutoPass License Server

APSC AutoPass Service Center

API Application Program Interface

REST Representational State Transfer

PD file Product Definition file

HTTP Hyper Text Transfer Protocol

HTTPS Hyper Text Transfer Protocol Secure