Working with Absolute APIs

www.absolute.com

April 2024

ABSOLUTE®

Working with Absolute APIs — Document revision: 9.0-0

Absolute Software Corporation
Suite 1400 Four Bentall Centre
1055 Dunsmuir Street
PO Box 49211
Vancouver, British Columbia
Canada V7X 1K8

Contents

Our APIs provide you with access to Absolute functionality and data without having to use the Secure Endpoint Console.

NOTE Depending on the permissions associated with your API token and the Absolute product licenses associated with your account, the Absolute APIs may not be available.

This document provides information about:

- Accessing and authenticating APIs
- Preparing requests for APIs
- Authenticating headers in a request
- Filtering and sorting
- Troubleshooting

IMPORTANT Absolute APIs support connections using Transport Layer Security (TLS) protocol version 1.2 only.

Accessing and authenticating APIs

To access the Absolute APIs, you use the Secure Endpoint Console to perform the initial setup, which includes assigning user roles and providing access to the console. You must then create and manage API tokens that are required for authentication and authorization.

API access

The URL you use to access the Absolute APIs depends on which URL you use to access the Secure Endpoint Console:

- If you use https://cc.absolute.com to access the console, use the following URL to access the API:

  https://api.absolute.com

- If you use https://cc.us.absolute.com to access the console, use the following URL to access the API:

  https://api.us.absolute.com

- If you use https://cc.eu2.absolute.com to access the console, use the following URL to access the API:

  https://api.eu2.absolute.com

NOTE Examples in this document use https://api.absolute.com

Prerequisites

There are two prerequisites before developers can use the Absolute APIs. The developer:

- must be assigned an Absolute user role. The role can be a default user role, or a custom user role defined by your organization.
- has at least one API token in the Secure Endpoint Console.

API tokens

NOTE The V2 APIs only support generated tokens.

A generated API token consists of two parts: token ID and secret key. The token ID is a random UUID, and the secret key is generated with a crypto-level random number generator.

Permissions

By default, the token has the same permissions as your assigned user role. This means that if you have access to certain functionality in the Secure Endpoint Console, the token you create has equivalent access in the API. If your user account is assigned to a single device group when the token is created, the token is assigned that device group. If your user account's device group assignment is updated in the future, the token isn't updated. The token is still assigned to the original device group.

When you create the token, you can modify the token's permissions. You can't assign a token permissions that your user role doesn't have, but you can remove permissions. For added security, limit the API token to the minimum permissions required for its intended use. Once the token is created, the permissions for the token can't be changed.

NOTE Only those permissions necessary for API calls can be added to an API token.

If the user account associated with the token is suspended or deleted, the token is no longer valid. If the user is deleted, the token is also deleted.

Permission requirements

To use an endpoint in the API, the token generally requires the same feature permissions that your user role requires to perform the action in the console. To use the Device Report API to return basic device information, the token requires the *View* permission for both Device Fields and Device reports. The token may also require more permissions depending on the parameters you want to return. For example, to see the `geoData` parameter, the token also needs to have the *View* and *Address-level view* permissions for Geolocation.

NOTE For all Reach endpoints, the token must have the *Run* permission for Reach Script.

Expiry

Newly created tokens must have an expiry date. By default, the expiry date is 90 days from the day the token is created. You can set the expiry date to be up to one year from the creation date. If you edit a token that doesn't have an expiry date, you are required to add an expiry date before you can save your changes. You can continue to use the token until 23:59:59 UTC on the day the token expires. Before a token expires, you can edit the expiry date in the Absolute console.

Token ID

The token ID is a random GUID-like string and is public information, like a user name. It is associated with the same role and device group as the Absolute user account.

Secret key

The secret key is a random sequence of bits and is private and sensitive information.

IMPORTANT Store this key securely, and do not share it.

Creating an API token

To create an API token containing a generated token ID and secret key:

1. Log in to the Secure Endpoint Console as user with the *Manage* permission for API credentials.
2. On the navigation bar, click **Settings** > **API management**.
3. Click **Create API token**.
4. In **Add title**, give the token a name.
5. [Optional] To help identify the token, click the field under the title and enter a **Description** for this request.
6. Select **Generate Token**.
7. Click the **Expiration** date field and set an expiration date at least one day in the future by doing one of the following:
   - Enter a date in YYYY-MM-DD format
   - Use the Calendar picker to select a date
   - Select one of the predefined expiry date ranges

   If you don't set the **Expiration**, the expiration date defaults to 90 days.
8. To change the token's permissions, select **View**, **Manage**, and **Other Actions** for each permission you want to add. Clear **View**, **Manage**, and **Other Actions** for each permission you want to remove. If you select **Manage** for a permission, **View** is automatically selected. If you don't change the permissions, the token has the same permissions as your assigned user role.

   IMPORTANT Token permissions can't be modified once the token is created.

   To include all the permissions assigned to your user role, scroll to the bottom of the Permissions section and click **Select All**. To remove all permissions, click **Clear**.

   NOTE Assigned device group is for your information only.
9. [Optional] Enter the IP addresses that you want to allow to access the APIs. Both IPv4 and IPv6 IP addresses in single and CIDR format are accepted. If no IP addresses are entered, the APIs can be accessed from any IP address. Approved IP addresses can be added to a token after it has been created.
   a. Enter or copy and paste one of the following in to **Approved IP Address**:
      - An individual IP address
      - A list of IP addresses separated by a space ( ), comma (,), semi-colon (;), or line break

   b. Click **Add** or press ENTER.

      IP addresses are listed below the entry field. Validation is done on each IP address. If validation fails, you see **Invalid IP address**. Do one of the following:
      - To delete the IP address, click ⊖ (**Remove**).
      - To edit the IP address, click in the IP address, make your changes, and press TAB or click away from the IP address. Validation is done on the updated IP address.

   Duplicate entries are ignored.

10. Click **Save**.

    After you click **Save**, none of the fields can be edited. To make changes, you need to edit the token.

11. From the Token Key Details section, do one of the following to capture the token information:

    - To copy the token information
      a. Click in the **Token ID** field or click **Copy** beside the token ID and paste the token ID to a text file.
      b. Click in the **Secret key** field or click **Copy** beside the secret key and paste the secret key to the text file.
      c. Save the file to a secure location on your computer.
    - To download the token ID and secret key, click **Download Token**. The token ID and secret key are downloaded in a `.token` file to your operating system's downloads folder. You can use a text editor, such as Notepad, to open the file.

    IMPORTANT If you close this dialog without downloading or copying the secret key for generated tokens, you cannot retrieve the information later. Record or save the secret key now, or you must delete this token and create a new one.

12. After you have captured the token information, click ✕ (**Close**).

On the API Token Management page, the new token is added to your list of tokens. An *API token updated* event is logged to Event History. For more information on token management, see Managing API tokens in the online help.

IMPORTANT It is imperative that you keep your secret keys secure. They are comparable to passwords—don't share them with anyone.

Preparing requests for APIs

You must use proper format and include your token in order to properly authorize your API request. To make an API request, you:

1. Create a canonical request
2. Create a signing string
3. Create a signing key
4. Create a signature
5. Add the authorization header

This document provides some basic examples. For more code samples, contact Absolute Technical Support (www.absolute.com/en/support).

Creating a canonical request

The canonical request looks like this:

```
CanonicalRequest =
  HTTPRequestMethod + '\n' +
  CanonicalURI + '\n' +
  CanonicalQueryString + '\n' +
  CanonicalHeaders +
  LowerCase(HexEncode(Hash(RequestPayload)))
```

The following table describes the parameters:

CanonicalRequest parameters

Parameter	Description
HTTPRequestMethod	All uppercase request methods: • `GET` • `POST` • `PUT` • `DELETE`
CanonicalURI	The Request path, excluding the hostname and query parameters Path segments are URL-encoded in case they contain spaces or other characters. Normalize the path using URI generic syntax ***Example*** URL: `https://api.absolute.com/v2/reporting/devices?$top=2` URI path: `/v2/reporting/devices` CanonicalURI: `/v2/reporting/devices`
CanonicalQueryString	The entire query string, URL-encoded If no query-string is present, use an empty string "" 1. Create a list of all arguments. 2. Sort arguments in ascending order; for example, 'A' is before 'a'. 3. If not already encoded, URI encode the parameter name and value using URI generic syntax. 4. Reassemble the list into a string. For each argument in argumentList: `{ str += argument.name + '=' + argument.value; if ! last argument str += '&'; }` ***Example*** The following is for the CanonicalQueryString for the top 10 results, skipping the first 20 results: 1. Create a list of arguments: • `$top=10` • `$skip=20` 2. Sort arguments in ascending order: • `$skip=20` • `$top=10` 3. URI enocde: • `%24skip=20` • `%24top=10`

Parameter	Description
	4. Reassemble the list into a string: • `%24skip=20&%24top=10`

CanonicalHeaders	Only a subset of headers is included
	Sample code:
	`CanonicalHeaders=""; //For each header in ProtectedHeaders`
	`{ CanonicalHeaders+= lowercase(header)`
	`+ ':'`
	`+ trimmed(header value)`
	`+ '\n';`
	`}`

Encoded hash of payload	Hash the entire body using SHA-256 algorithm, HexEncode, and apply lowercase If there is no payload, use an empty string

Example basic canonical request

The following request has no query parameters:

```
GET
/v2/reporting/devices

host:api.absolute.com
content-type:application/json
x-abs-date:20170926T172032Z
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example canonical request with one query parameter

The following canonical request has one query parameter:

- `substringof('760001', esn) eq true`
  URL encoded: `substringof%28%2760001%27%2C%20esn%29%20eq%20true`

```
GET
/v2/reporting/devices
%24filter=substringof%28%2760001%27%2C%20esn%29%20eq%20true
host:api.absolute.com
content-type:application/json
x-abs-date:20170926T172213Z
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Example canonical request with two query parameters

The following canonical request has two query parameters:

- `substringof('760001',esn) eq true`
  URL encoded: `substringof%28%2760001%27%2C%20esn%29%20eq%20true`
- `substringof ('2700000', esn) eq false`
  URL encoded: `substringof%28%2760000%27%2C%20esn%29%20eq%20false`

```
GET
/v2/reporting/devices
%24filter=substringof%28%2760001%27%2C%20esn%29%20eq%20true%20and%20substringof%28%2760000%27%2C%20esn%29%20eq%20false
host:api.absolute.com
content-type:application/json
x-abs-date:20170926T172255Z
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Creating a signing string

The signing string uses this format:

```
StringToSign =
 Algorithm + \n +
 RequestDateTime + \n +
 CredentialScope + \n +
 HashedCanonicalRequest
```

Parameters

The following table shows descriptions and examples of the parameters of the signing string:

StringToSign parameters

Parameter	Description
Algorithm	The string used to identify the algorithm ***Example*** `ABS1-HMAC-SHA-256`
RequestedDateTime	The date and time (in UTC) from **X-Abs-Date** Format: ***<YYYY><MM><DD>***T***<HH><MM><SS>***Z ***Example*** `20170926T172032Z`

Parameter	Description
CredentialScope	The CredentialScope is defined in three parts: 1. the date (in UTC) of the request Format: ***YYYYMMDD*** 2. region or data center (*must* be in lowercase) Possible values: • `cadc` • `usdc` • `eudc` NOTE Each data center has a unique URL. 3. version or type of signature Always **`abs1`** ***Example*** `20170926/cadc/abs1`
HashedCanonicalRequest	The hashed, hex-converted, and lowercase value of the canonical request. ***Example*** `63f83d2c7139b6119d4954e6766ce90871e41334c3f29b6d64201639d273fa19`

Example StringToSign

The following StringToSign is based on the example values in from the previous table:

```
ABS1-HMAC-SHA-256
20170926T172032Z
20170926/cadc/abs1
63f83d2c7139b6119d4954e6766ce90871e41334c3f29b6d64201639d273fa19
```

Creating a signing key

HMAC-SHA256 is used for authentication.

The following table shows descriptions of the inputs used to create a signing key:

Signing key inputs

Input	Description

`kSecret=UTF8.GetBytes("ABS1"+secret)`
Alternative sample pseudocode:
`kSecret=UTF8.GetBytes(String.Concatenate("ABS1",secret))`
	The kSecret value is calculated by concatenating the static string "ABS1" with the value of the secret key from your API token and then encoding the resulting string using UTF8.
	The secret is the secret key value from the token that you created in the Secure Endpoint Console.

`kDate=HMAC(kSecret,Date)`	The date (in UTC) of the request Format: ***<YYYY><MM><DD>*** The result is a byte array.
`kSigning=HMAC(kDate,"abs1_request")`	Use the binary hash to get a pure binary kSigning key NOTE Do not use a hexdigest method. The result is a byte array.

Creating a signature

As a result of creating a signing key, kSigning is used as the key for hashing. The StringToSign is the string data to be hashed.

The signature looks like this:

```
signature = lowercase(hexencode(HMAC(kSigning, StringToSign)))
```

Parameters

The following table shows describes the parameters.

Signature parameters

Parameter	Description
kSigning	The byte array that was created from the signing key
StringToSign	The signing string

Example of signing the string

This example shows the resulting signature for a request which is then used in the authorization header.

```
Signature=e15b64a4f91a0e53c2f91a6f52756a74bc21e6f175795cbf85bc15e8ef32aab5
```

Adding the authorization header

Use the standard HTTP Authorization header.

```
Authorization: <algorithm> Credential=<token id>/<CredentialScope>,
SignedHeaders=<SignedHeaders>, Signature=<signature>
```

Parameters

The following table shows descriptions and examples of the parameters of the authorization header.

Authorization header parameters

Parameter	Description
Authorization	The string used to identify the algorithm ***Example*** `ABS1-HMAC-SHA-256`
Credential	The token ID ***Example*** `cc2423f2-cc28-48a6-9dce-a268d5e3cd01`
CredentialScope	The CredentialScope is defined in three parts: 1. The date (in UTC)of the request Format: ***<YYYY><MM><DD>*** 2. Region or data center (*must* be in lowercase) Possible values: • `cadc` • `usdc` • `eudc` NOTE Each data center has a unique URL. 3. Version or type of signature ***Example*** `20210926/cadc/abs1`

Parameter	Description
SignedHeaders	Semi-colon (;) delimited list of lowercase headers used in CanonicalHeaders ***Example*** `host;content-type;x-abs-date`
Signature	The fully calculated resulting signature from the signing key and the signature ***Example*** `e15b64a4f91a0e53c2f91a6f52756a74bc21e6f175795cbf85bc15e8ef32aab5`

Example authorization header

The following authorization header uses the values from the previous table.

```
1 Authorization: ABS1-HMAC-SHA-256 Credential=cc2423f2-cc28-48a6-9dce-
  a268d5e3cd01/20170926/cadc/abs1, SignedHeaders=host;content-type;x-abs-date,
  Signature=e15b64a4f91a0e53c2f91a6f52756a74bc21e6f175795cbf85bc15e8ef32aab5
```

NOTE There is a space after each comma in the authorization header. It may not appear if you use copy and paste.

Authenticating headers in a request

We include only a small subset of HTTP headers in a request to minimize the possibility of proxies modifying them in transit.

You must use the following headers for all of your API requests.

Authentication headers

Header	Description
Host	The domain name of the server where the request is sent ***Example*** `api.absolute.com`
Content-Type	The media type of the resource ***Example*** `application/json`
X-Abs-Date	The automatically generated header that indicates the time (in UTC) the request was made Format: ***<YYYY><MM><DD>***T***<hh><mm><ss>***Z. ***Example*** `20210926T172032Z`

Header	Description
Authorization	The HTTP authorization header Format: ***<algorithm>*** Credential=***<token id>/CredentialScope***, SignedHeaders=***<SignedHeaders>***, Signature=***<signature>*** ***Example*** `ABS1-HMAC-SHA-256 Credential=cc2423f2-cc28-48a6-9dce-a268d5e3cd01/20210926/cadc/abs1, SignedHeaders=host;content-type;x-abs-date, Signature=e15b64a4f91a0e53c2f91a6f52756a74bc21e6f175795cbf85bc15e8ef32aab5` NOTE There is a space after each comma in the authorization header. It may not appear if you use copy and paste.

Filtering and sorting

Absolute uses a subset of query options from Open Data Protocol (OData) for filtering and sorting. OData version 1 and 2 are supported. OData query parameters must be alphabetized and URI encoded.

For more information about OData, see: https://www.odata.org/documentation.

The applicable OData system query options are:

- `$filter`
- `$orderby`
- `$select`
- `$skip`
- `$top`

`$filter`

The `$filter` system query option filters items included in the response with the specified expression.

Operators supported by $filter are described in the following tables:

- Logical operators supported by `$filter`
- Grouping operators supported by `$filter`
- String functions supported by `$filter`

The examples in the table use the Device Reporting API. For example, `GET /v2/reporting/devices?$filter=agentStatus eq 'A'`.

Logical operators supported by `$filter`

Logical operator	Description

Eq	Equal
	Examples
	To view a list of all devices with an Active status:
	• $filter=agentStatus eq 'A'
	• URL encoded:
	`%24filter=agentStatus%20eq%20%27A%27`
	To view a list of all devices that are currently frozen:
	• $filter=dfStatus.statusCode eq 'FRZN'
	• URL encoded:
	`%24filter=dfStatus.statusCode%20eq%20%27FRZN%27`
	To view a list of all devices that have 1734 in their ESN (Identifier):
	• $filter=substringof('1734',esn) eq true
	• URL encoded:

Ne	Not equal
	Examples
	To view a list of all devices with a status that isn't Active:
	• $filter=agentStatus ne 'A'
	• URL encoded:
	`%24filter=agentStatus%20ne%20%27A%27`
	To view a list of all devices with a status that isn't Disabled:
	• $filter=agentStatus ne 'D'
	• URL encoded:
	`%24filter=agentStatus%20ne%20%27D%27`

Gt	Greater than ***Example*** To view a list of all devices with greater than 1 GB (1073741824 bytes) of available physical ram: • $filter=availablePhysicalRamBytes gt 1073741824 • URL encoded: `%24filter=availablePhysicalRamBytes%20gt%201073741824`

Logical operator	Description
Ge	Greater than or equal ***Example*** To view a list of all devices with greater than or equal to 1 GB (1073741824 bytes) of available physical ram • $filter= availablePhysicalRamBytes ge 1073741824 • URL encoded: `%24filter=availablePhysicalRamBytes%20ge%201073741824`
Lt	Less than ***Examples*** To view a list of all devices with less than 1 GB (1073741824 bytes) of available physical ram • $filter=availablePhysicalRamBytes lt 1073741824 • URL encoded: `%24filter=availablePhysicalRamBytes%20lt%201073741824` To view a list of all devices that have a connection date/time less than January 1, 2021 at midnight (yyyy-mm-ddThh:mm:ssZ): • $filter=lastConnectedUtc lt datetime'2021-01-01T00:00:00Z' • URL encoded: `%24filter=lastConnectedUtc%20lt%20datetime%272021-01-01T00%3A00%3A00Z%2`
Le	Less than or equal ***Examples*** To view a list of all devices with less than or equal to 1 GB (1073741824 bytes) of available physical ram • $filter=availablePhysicalRamBytes le 1073741824 • URL encoded: `%24filter=availablePhysicalRamBytes%20lt%201073741824` To view a list of all devices that have a connection date/time less than or equal to January 1, 2021 at midnight (yyyy-mm-ddThh:mm:ssZ): • $filter=lastConnectedUtc le datetime'2021-01-01T00:00:00Z' • URL encoded: `%24filter=lastConnectedUtc%20le%20datetime%272021-01-01T00%3A00%3A00Z%27`

Logical operator	Description
And	Logical and ***Example*** To view a list of all devices with less than 1 GB (1073741824 bytes) and more than 500 MB (524288000 bytes) of available physical ram • %24filter=availablePhysicalRamBytes lt 1073741824 and availablePhysicalRamBytes gt 524288000 • URL encoded: `%24filter=availablePhysicalRamBytes%20lt%201073741824%20and%20availablePhysicalRamBytes%20gt%20524288000`
Or	Logical or ***Examples*** To view a list of all devices with less than 1 GB (1073741824 bytes) of available physical ram or less than 1 GB (1073741824 bytes) of available virtual ram • $filter=availablePhysicalMemroyBytes lt 1073741824 or availableVirtualMemoryBytes lt 1073741824 • URL encoded: `%24filter=availablePhysicalRamBytes%20lt%201073741824%20or%20availableVirtualMemoryBytes%20lt%201073741824` To view a list of devices with '1734' in IEMI, serial number, or a custom device field • $filter=substringof('1734', imei) or substringof('1734', serial) or (cdf ne null and substringof ('1734',cdf.l1Z41xJ2S2GbreeK8pChkg)) • URL encoded: `%24filter=substringof%28%271734%27%2Cimei%29%20or%20substringof%28%271734%27%2Cserial%29%20or%20%28cdf%20ne%20null%20and%20substringof%28%271734%27%2Ccdf.l1Z41xJ2S2GbreeK8pChkg%29%29`
Not	Logical negation ***Example*** To view a list of devices with a username that doesn't contain LPTP • $filter=not substringof('LPTP',username) • URL encoded: `%24filter=not%20substringof%28%27LPTP%27%2Cusername%29`

Grouping operator supported by `$filter`

Grouping operator	Description
( )	Precedence grouping ***Example*** /Products?$filter=(Price sub 5) gt 10

Functions can also be used with `$filter`.

NOTE You can use a NULL literal in comparisons as ISNULL or COALESCE operators are not defined.

String functions supported by `$filter`

String function	Description
bool substringof (string p0, string p1)	Returns records where the value of the parameter in *string p1* contains *string p1* ***Example*** To view a list of all devices where the ESN contains the string '1734' • $fitler=substringof('1734',esn) • URL encoded: `GET /v2/reporting/devices?%24filter=substringof%28%271734%27%2Cesn%29`
bool endswith (string p0, string p1)	Returns records where the value of the parameter in *string p1* ends with *string p0* ***Example*** To view a list of all devices where the ESN ends with the string '1734' • $filter=endswith('1734',esn) • URL encoded: `GET /v2/reporting/devices?%24filter=endswith%28%271734%27%2Cesn%29`

$orderby

The `$orderby` system query option sorts a resulting list of data as you instruct.

To sort the Device Report so that devices with the most current lastUpdatedUtc dates show first in the list:

- $orderby=lastUpdatedUtc desc
- URL encoded:

  `GET /v2/reporting/devices?%24orderby=lastUpdatedUtc%20desc`

$select

The `$select` system query option requests a specific set of properties.

To see only the manufacturer, model, and serial number attributes of your devices:

- $select=systemManufacturer,systemModel,serial
- URL encoded:

  `GET /v2/reporting/devices?%24select=systemManufacturer%2CsystemModel%2Cserial`

$skip

The `$skip` system query option requests the number of items to be excluded from the result.

To return the second page of results when data is returned in batches of 20:

- $skip=20&$top=20
- URL encoded:

  `GET /v2/reporting/devices?%24skip=20&%24top=20`

NOTE Use with the $top query option to paginate your results.

$top

The `$top` system query option requests the number of items to be included in the result.

To limit the number of records returned to the first 10:

- $top=10
- URL encoded:

  `GET /v2/reporting/devices?%24top=10`

Pagination

For the Device Reporting API and the Software Reporting API, you can use the `$skip` and `$top` query parameters together to paginate your results. The `$top` query specifies the number of results to be returned. The `$skip` query specifies the number of results that are excluded from the result.

Example

Using the Device Reporting API, you want to get the first 60 results, returned in batches of 20.
To get the first 20 results, results 1 to 20, you only need to use the `$top` query.

```
GET /v2/reporting/devices?%24top=20
```

To get the next 20 results, results 21 to 40, you need to skip the first 20, which were returned in the last request. Use `$skip` to exclude the first 20 results, and `$top` to get the next 20 results.

```
GET /v2/reporting/devices/%24skip=20&%24top=20
```

To get the final 20 results, results 41 to 60, you need to skip the first 40, which were returned in the first two requests. Use `$skip` to exclude the first 40 results, and `$top` to get the next 20 results.

```
GET /v2/reporting/devices/%24skip=40&%24top=20
```

Troubleshooting

The following table lists some HTTP status codes and describes the corresponding errors that you may encounter when using Absolute APIs.

HTTP status code errors

Status code	Possible error	Mitigation
401	Incorrect HTTP method	Ensure that the method used in the calculation matches the actual request method.
	Incorrect HTTP method case	In the calculation, the HTTP method must be all uppercase; for example, `GET`, `POST`, `PUT`, `DELETE`.
	Incorrect X-Abs-Date value or format	Ensure that the date value is accurate, is in the UTC timezone, and is in the ***<YYYY><MM><DD>***T***<HH><MM><SS>***Z format.
	Incorrect Query Parameters encoding	Ensure that the query parameters are URL-encoded when creating the CanonicalRequest. For example, the query parameter `$top=15` would be encoded as `%24top=15`.
403	The tokenID belongs to a user that does not have permission to access the URL.	Ensure that the user who created the token has the appropriate permissions.

If you are unable to resolve the errors listed above, enable **Authentication Debugging** in the Secure Endpoint Console on the API Token Management page. After that, repeat the failed API requests to log the details. You can then contact Technical Support with the details collected in the log. Ensure to include the following:

- tokenID
- CanonicalRequest
- X-Abs-Date
- Signature