Public APIs

Veritas SaaS Backup

The truth in information.

Table of Contents

Overview

This document describes the world accessible public API to the Veritas SaaS Backup system. This API is used by clients (PC backup client, iOS file access client and Web file access client) for all operations on files as well as user settings.

The API is a RESTful API accessed over HTTPS is available at the regional URL's:

Europe: https://eu.saasbackup.veritas.com
Americas: https://us.saasbackup.veritas.com
Australia: https://au.saasbackup.veritas.com

All request and response document schemas are documented in an RNC-like syntax (see www.relaxng.org). The API generally follows these conventions:

- Numbers are decimal
- All dates, timestamps and periods follow a subset of ISO8601 and use UTC ("Zulu") time zone
- All documents are UTF-8 encoded

Some method/URI combinations will require user authentication, and some can be performed unauthenticated. This is documented on the individual endpoints.

Back-end systems overview

The Veritas SaaS Backup back-end is comprised of a number of individual systems which - in combination - provide the described back end API. Some of the components provide subsets of the API - like for example the Object Store back end provides the data access API. The intention is to allow us to construct relatively simple individual components which are then combined to form the full system. It should also make it less painful to replace individual components of the back-end system over time, if or when this is deemed necessary.

The "Proxy" is the world-accessible end point to which the clients will connect. The proxy implements a major part of the API itself and forwards the remaining requests to the relevant back-end systems (such as, for example, object store requests to the object store servers).

The proxy has a secondary function; it holds a SSL certificate to our site name and provides termination of client HTTPS connections. It then forwards the protocol requests internally on our back-end systems using plain HTTP. This prevents us from having to deal with SSL certificates and CPU overhead on our "simple" component back end systems.

The proxy function will actually be served by a number of servers by means of a DNS record that resolves to a number of IPs. This pool of hosts will provide load balancing and fault tolerance, since there is no state on the proxy - so clients can (and will) hop between proxies as they issue requests against the back end.

Client developer overview

When developing a client, whether it is a CRM integration, a read-only data access client or a full-fledged backup client, you will be connecting to a RESTful API over HTTPS running from a DNS name that points to several IP addresses.

You can connect to any of the IPs, as they point to servers all serving the same API with the same underlying data.

We use HTTP/1.1 and support both persistent connections and pipelining. It would be prudent for the client to use persistent connections, to limit the number of reconnects against the servers.

Mapping external system account IDs to Veritas account IDs

When creating an account, it is possible to supply an "external_id"; this id will be stored in the Veritas' account database and can be used to later reference the account.

Every API end-point under /users/{guid} will accept /users/extrn-{external_id} instead of /users/{guid}.

Thus, an account could be created with this request:

```
POST /users/add227e7-7f01-4641-803a-864a920aa816/users HTTP/1.1
host: foo
authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK
content-length: 116
<user_create>
 <login>h@qf</login>
 <password>there</password>
 <external_id>4201</external_id>
</user_create>
```

And the device list for the account could then be retrieved using this request:

```
GET /users/extrn-4201/devices HTTP/1.1
host: foo
authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQK
```

Notice how the GUID is simply replaced with "extrn-" followed by the external_id supplied when the account was created.

If an external id is given that cannot be resolved, the request will simply fail with a 404 error.

Any request using external ids must be authenticated as the partner account that created the user because external ids are unique within the partner.

Important note about connecting to the API

The API is available on "https://saasbackup.veritas.com". This DNS name, however, resolves to multiple IP addresses. The purpose of this, is, to provide fault tolerance and load balancing. In a scenario were one datacenter is lost, the IPs served by proxies in that datacenter will simply be overtaken by proxies on a remaining datacenter. Since connecting clients will connect to an IP address selected randomly among the ones resolved, client load will effectively distribute itself evenly among the actual proxies without further complicated logic.

Important note about access tokens

When a user account is created, we create one access token for it also. This is the user selected username and a user selected password. This information must never be stored on any client device - not in the iOS app, not in a cookie in a browser, not in the PC client configuration, not anywhere. The user's username and password must NEVER be stored.

When a client connects to the API, it should have a client specific access token. The client will store this persistently if possible (configuration file, cookie, local database, registry, ...). That way, the client can use its token to access the API without prompting the user for a username and password. Further, the user can easily delete the token if the device to which the token belongs, is stolen or lost. And, finally, if the user decides to change his password, he will not have to update it on every single device, since the device access tokens will remain unchanged.

If a device does not yet have a device access token, one must be created. The procedure for this is as follows:

1. Prompt the user for username and password (U and P)
2. Generate a device username and device password (dU and dP)
3. Send a POST /tokens/ message to create a token with a short client device description (Jörgen's iPhone for example) and the dU and dP. This POST request will have to be authenticated using HTTP Basic authentication using the user supplied U and P.
4. Store the dU and dP in local storage (registry, config file, database, cookie, ...)
5. For all future API access, use HTTP Basic authentication with our dU and dP.

Thus, the user supplied username and password is only transmitted over the network once and it is never again used and therefore need not be stored.

Protocol usage overview

If the client does not have an access token, it will ask the user to provide username and password. The client will then issue an HTTPS request using HTTP Basic authentication (with the supplied username and password) in order to request an access token. The client will only store the access token, never the user-supplied username and password.

Assuming the client has an access token (a specific "username" (GUID) and "password" (large key)), it will authenticate its HTTPS requests to the proxy using HTTP Basic authentication.

Protocol usage examples

Listing available devices

Try this in your browser:

https://saasbackup.veritas.com/users

The browser will ask you for credentials (because we use standard HTTP authentication) - you should have received credentials for testing.

You will be presented with the id of the user(s) add this id to the end of the URL:

https://saasbackup.veritas.com/users/{user_id}

This will give you the Product for this user and the id for the parent of the user.

Adding devices will give you the list of devices for this user.

https://saasbackup.veritas.com/users/{user_id}/devices/

Listing backup history for device

Try this in your browser:

https://saasbackup.veritas.com/users/{user_id}/devices/{device_guid}/history/

This gives you a document with root ids for various backup sets.

Creating a new user

In order to create a user, one would use the "POST /users/{guid}/users" endpoint, which allows for the creation of a sub-account under an existing account. The "{guid}" in the URI would be the guid of the partner account under which the new user account is to be created.

The following XML document can be sent using, for example, the curl utility to the saasbackup.veritas.com API servers.

```
<user_create>
 <login>joe@test123.com</login>
 <external_id>id3367-987655</external_id>
 <product>g59weh-q6vu7r-dr1tys</product>
 <email>joe@test123.com</email>
 <language>da</language>
</user_create>
```

The above request will create an account with the login "joe@test123.com" under the partner with guid ucjkdp-w8xrwg-vakfv5. The account will also have its e-mail address set to joe@test123.com (technically the login does not need to be an e-mail address, but current limitations in the web UI restrict logins to be e-mail addresses).

We use the "external_id" to associate a partner-internal customer account number - for example from an existing CRM system. This account can then always be referenced by the partner using this external id, instead of the login - so that if the login is changed at a later day, the account is still uniquely identified by the external id.

Many more elements are possible, and several of those used here are not required - please consult the actual API endpoint documentation for the full details.

Here is a command-line example of how to send the document - in this example we use the "curl" tool on a Mac:

```
$ curl -vd "<user_create><login>joe@test1234.com</login><external_id>id3367-
9876556</external_id><product>g59weh-q6vu7r-
dr1tys</product><email>joe@test1234.com</email><language>da</language></user_create>" h
ttps://JJ81XInS:JJ81XInS@saasbackup.veritas.com/users/ucjkdp-w8xrwg-vakfv5/users
*   Trying 212.97.129.25...
* Connected to saasbackup.veritas.com (212.97.129.25) port 443 (#0)
* TLS 1.2 connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate: *.saasbackup.veritas.com
* Server certificate: GeoTrust SSL CA - G3
* Server certificate: GeoTrust Global CA
* Server auth using Basic with user 'JJ81XInS'
> POST /users/ucjkdp-w8xrwg-vakfv5/users HTTP/1.1
> Host: saasbackup.veritas.com
> Authorization: Basic Sko4MVhJblM6Sko4MVhJblM=
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Length: 192
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 192 out of 192 bytes
< HTTP/1.1 201 Created
< access-control-allow-headers: authorization, authentication-failure-code, location
< access-control-allow-origin: *
< access-control-expose-headers: locationx-total-count
< location: /users/k62u54-3r5ofg-3sml9z
< content-length: 0
< date: Tue, 24 May 2016 11:23:58 GMT
< connection: keep-alive
<
* Connection #0 to host saasbackup.veritas.com left intact
$
```

As can be seen, the API responds with "201 Created."

On our production environment, for the purpose of this test, we have a partner account with the public credentials (username JJ81XInS, password JJ81XInS). This partner account and these credentials can be used for such API tests. Please note, however, that public credentials are intended only for sign-up form user creation (where credentials will be publicly available), and therefore these credentials cannot be used to modify or even enumerate existing accounts.

Workarounds for known browser issues

AJAX developer API workaround for browsers that intercept 401 errors

It appears that there is no general way to properly handle 401 errors in AJAX applications, because the browsers intercept this error (as the only error) and force the display of an authentication pop-up dialogue.

The only "solution" to this problem is to deviate from good practice and use an alternative error code. So, when the API wants to report an authentication failure which the HTTP standard has assigned code 401 for, then the "solution" when working on the browsers of today, is to use for example code 403 "forbidden" or some other code. Since current browsers force us away from the correct error code, it doesn't matter much which of the wrong codes we use - it can be argued though that 403 is less wrong than any of the others... At least 403 has something to do with access rights.

The API supports the following header; "authentication-failure-code" which can contain an integer in the range [400;499].

If this header is present, the status code it specifies will be used by the API when reporting an authentication failure instead of the correct 401 code.

This will allow AJAX applications to properly handle authentication errors, by requesting that a non-401 error code is sent back in situations where a 401 error code was actually the appropriate response. This is a workaround for a common browser deficiency that does not impose any downsides for non-browser clients; everyone else can still expect to get proper standard HTTP codes, unless a non-standard code is explicitly requested in this header.

Javascript developer API workaround for poor EventSource framework

The Javascript EventSource framework (the built-in Server-Sent Event client in Javascript) does not use the same default authentication as the XMLHTTPRequest framework does; therefore, a browser that successfully sends XMLHTTPRequests against the API may fail EventSource use against queues that require authentication. No provisions for supplying custom headers exist in the standard Javascript EventSource either - it is therefore not possible to supply an authentication header directly, nor indirectly by accessing the underlying XMLHTTPRequest.

In absence of an Authentication header in the request, our API checks for an "authentication" option. If such option exists, its value is treated precisely like the value of the authentication header. For example:

```
GET /vqueuees/evlog
Authentication: Basic cm9vdDoqKioqKg==
...
```

The above could alternatively be replaced by:

```
GET /vqueues/evlog?authentication=Basic%20cm9vdDoqKioqKg%3D%3D
```

```
...
```

Sending authentication headers as URI options is terribly bad style and this should never be employed where authentication headers are possible. But in rare cases, such as with the current Javascript EventSource framework, passing the authentication header as an option can be the least bad solution.

Security wise, headers and URI options are no different - both require HTTPS to be safe, and both are safe using HTTPS.

Account services API

This section of the API is used for changing basic user information such as, for example, account name, e-mail address, username, etc.

The API is used directly by sign-up pages, CRM integrations, and by Veritas clients and web UI for various aspects of user account management.

Method GET /users/

Response

The response document adheres to the following schema:

```
element user {
 element id { text }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	FAIL

Method POST /users/{parent_account_id}/users

Any user can create sub- user accounts. However, this is most commonly used by partners. For example, a partner provisioning user through APS will have a partner user account the APS system will authenticate to; all end-user accounts (or company accounts) will be created under that partner, using this API.

The parent_account_id given must be either the authenticated client account or a child account of the authenticated client.

So, if Partner A provisions a customer company B, they can actually also provision a company employee C which is a child of B, by supplying the account id of B as the parent_account_id of C.

Request

The request body document must adhere to the following schema:

```
element user_create {
 element login { text }        # If login contains a '@' and no separate email is
supplied, login is used for primary email too
 & element password { text }?
 & element external_id { text }?
 & (element type { text }     # AnonymousParent, PartnerParent, Administrator, User,
Device, System; or User if not specified
    | element acl { text })? Expanded list of ACLs
 & element product { text }? # id of product assigned to account
 & element fullname { text }?  # If supplied, primary contact full name
 & element email { text }?     # If supplied, primary contact e-mail address
 & element country { text }?   # ISO 3166-1 alpha-2
 & element zipcode { text }?   # If supplied, primary contact zipcode
 & element language { text }?  # RFC 1766
 & element phone { text }?     # If supplied, primary contact phone
 & element attributes {
  element attribute {
   element name { text }
   element value { text }
  }*
 }?
 & element captcha { text }?   # If supplied, captcha response token, required if
Capchaless ACL is missing
}
```

The external_id field has no use inside the Veritas SaaS Backup back-end, it is solely a utility field for use by external provisioning systems

The login and password supplied will be used to create a User type access token tied to the newly created account. If no password is supplied, a very large random password (unguessable) is set, and a temporary token is created - this intended purpose for this is, that the welcome mail can include a reset_password link, allowing the user to "activate" his account by choosing a password.

The productid must be an id of a product created in the portfolio on the creating account or on a parent thereof.

The location header will contain the URI of the newly created user on the form /users/{new-user-id}

A primary contact record will be created for the created account, holding the supplied login as the e-mail address and, if supplied, the fullname.

Note on conflicts

User creation can conflict on either a duplicated login (logins are unique across the system), or a duplicated external_id (external_ids are unique under the parent). In both cases a 409 response is returned with a descriptive text in the body.

However, we also add a special header, named "conflict", which is set to either "login" or "external_id", depending on which of the two fields caused the conflict. In case both fields conflict, the header only contains one of them (the first one to cause the conflict - which in the current implementation is the login). This was added to ease generation of correct user feedback in user interfaces using this API.

ACL

Public	**Partner**	**Admin**	**User**	**Device**	**System**	**NONE**
OK	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}

This request is used by the user himself or by a parent account to this user account, to retrieve the user settings.

Response

The response body document adheres to the following schema:

```
element user {
 element enabled { boolean }
 & element created { iso8601-datetime }
 & element product { text }?
 & element parent { text }?
 & element deletion-deadline { iso8601-datetime }? # time after which user is wiped
from the database
 & element subscribed { boolean }
 & element external_id { text }?
}
```

Note, the product field reports the product set directly on this account. The account may have no product set, in which case it inherits its product from the parent (and so on recursively). This API endpoint reports what is set directly on the account, not what may be inherited. In order to deduce which product is in effect on the account, you need to traverse to the parent (and so on recursively) until a product is found.

Special nobl header

If the "nobl" header is supplied, no call to the business logic will be issued to resolve which product is assigned on the account; therefore the product element will be missing from the return document.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /users/{user-id}

This request is used by the user himself or by a parent account to this user account, to update the user settings.

Request

The request body document must adhere to the following schema

```
element user_update {
 element enabled { boolean }?
 & element subscribed { boolean }?
 & element product { text }? # empty value removes product assignment from the account
 & element external_id { text }?
 & element grace-expires { iso8601-datetime or empty }?
 & element parent { text }? # GUID of new parent account
}
```

This allows an account to be enabled or disabled, and it allows the product to be changed and the external_id to be changed. To clear the product, supply an empty string for the product.

If a user account is disabled, authentication will fail on that account.

Only System type tokens can set the grace-expires. If the grace-expires element is empty, grace time is un-set on this account.

If the parent element is set, this will move the account to the new parent GUID. Only an administrator or partner token can move an account, and the parent can only be set to an account at or beneath the authenticated account.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	OK	FAIL

Method DELETE /users/{user-id}

This request is used to delete a user account and everything associated with that account including access tokens, backup history, and all child accounts.

If delete-retention-period resource is set than account will be scheduled for deletion instead of deleting directly and all associated with that account will be saved till this period expires.

The initiating peer must be authenticated as either the user to be deleted or a parent of that user.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /users/{user-id}/revive

This method is used to resurrect scheduled for deletion user account.

Response

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/resources

This request is used to retrieve the resource consumption of the given user id including all sub-accounts. Important thing about this request: usage is returned only if product is assigned to the account directly and not available to subaccounts.

There's a special header though: 'x-total-usage'. Setting the value of it to "true" forces the API to return total usage of all accounts which belong to current product assignment.

Response

The response document will adhere to the following schema

```
element resources {
 element resource {
  element evaluated { iso8601-timestamp }? # time of last evaluation
& element name { text }
& element type { text }  # integer, boolean, period
& element unit { text }? # textual unit of numerical unit
& element limit { text }? # optional for integer and period, cannot be set for boolean
types
& element usage { text }? # sum of total use on product instance
& element violated { boolean }? # whether or not the limit is violated
& element grace-expires { iso8601-timestamp }? # If a resource is on grace, this marks
grace expiry
 }*
}
```

There will be a resource record for every resource that exists on the currently valid configuration of the product that this user is using.

However, we may not always have a utilization record for a resource. For example, on a fresh account with no backups, we may not have any data history yet. In that case, the 'evaluated' element will be missing (telling us that consumption was simply not evaluated for this resource).

Usually, though, both limit, usage and violated elements will be present along with the evaluated element.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/product

This request will retrieve the details of the product that is effectively assigned to this account.

An account may or may not have a product assigned to it. This can be determined whether a product is available in the /users/{guid} document.

If no product is assigned to this account, it will simply use resources from the product assigned to its parent (and so on recursively).

This request will return 404 if no product is assigned to this account or any ancestor account.

The grace-expires element is present only if the resource limit set on the product has been violated or if a provisions limit has been violated. If the grace-expires element is present, user agents should warn the user about this fact.

Technically, in the backend, we maintain a grace timer for each account which is used for provisions-based violations (which affect only the one account for which the provisions was violated) and a grace timer for the product instance which is used for product instance resource violations (which affect all accounts using the same instance). The grace-expires value returned here, is the minimum value of those two timers if both are set, or the set timer if only one is set.

Response

```
element product {
 element id { text }?       # only missing if nobl header is given
 & element name { text }?   # only missing if nobl header is given
 & element grace-expires { iso8601-datetime }?  # If we are on grace, this marks grace
expiry
}
```

Special nobl header

If the "nobl" header is supplied, no call to the business logic will be issued to resolve product id and name, and therefore those elements will be missing from the return document.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET|PUT /bl/partner/{partner-guid}/messages/languages

Returns a list of languages configured by account partner. Performs recursive search to the first partner account defining languages.

```
element languages {
 element language { lang }*   # Subset of RFC5646 langtag; language "-" region
                              # ISO639 language dash ISO3166-1 region code
                              # e.g. en-GB, da-DK, etc.
}
```

Method GET /users/{user-id}/contacts

This request will return all contacts registered for the given user.

Response

```
element contacts {
 element type { text }*
}
```

We only allow one contact of each type, and the response document will list all contact types (and therefore all contacts) registered for the given user.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/contacts/{contact-type}

This request will return the given contact for the given user.

Response

```
element contact {
 element type { text }
 & element email { text }?
 & element companyname { text }?
 & element fullname { text }?
 & element phone { text }?
 & element street1 { text }?
 & element street2 { text }?
 & element city { text }?
 & element zip { text }?
 & element state { text }?
 & element country { text }?
 & element language { text }?
}
```

We support two types of contacts right now:

- "p" - short for "primary", which is the primary contact information of the account.
- "s" - short for "support". This is used only by partners. The e-mail address supplied here is used as the "from:" address in e-mails sent to child accounts of that partner

For primary contacts, even though both email, companyname and fullname are optional, at least the e-mail should be present.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method POST /users/{user-id}/contacts

This request allows adding a contact to a user

Request

Currently we only support "primary" contacts, or type "p" contacts. The type field must for now contain a "p".

```
element contact {
 element type { text }
 & element email { text }?
 & element companyname { text }?
 & element fullname { text }?
 & element phone { text }?
 & element street1 { text }?
 & element street2 { text }?
 & element city { text }?
 & element zip { text }?
 & element state { text }?
 & element country { text }?
 & element language { text }?
}
```

The country code must be a valid ISO 3166-1 alpha-2 code.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Response

The response will contain a location header with the URI of the newly created contact.

Method PUT /users/{user-id}/contacts/{contact-type}

This request allows editing details of a given contact.

Request

Any field omitted in the request document will be erased from the given contact, so that a GET on this same contact will return exactly the fields included in this request plus the contact type.

```
element contact {
 element email { text }?
 & element companyname { text }?
 & element fullname { text }?
 & element phone { text }?
 & element street1 { text }?
 & element street2 { text }?
 & element city { text }?
 & element zip { text }?
 & element state { text }?
 & element country { text }?
 & element language { text }?
}
```

The country code must be a valid ISO 3166-1 alpha-2 code.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user-id}/users

This method will provide the list of direct sub-accounts for the specified user id.

The provided user-id must be the authenticated client user or a descendent thereof.

Response

The response document adheres to the following schema

```
element users {
 element id { text }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user-Id}/users?filter={filter}&max_depth={1..}&limit={1..}&offset={0..}&order_by_name={0|1}&reverse_order={0|1}&all={0|1}

Extended version of above-mentioned request to get the list of sub-accounts for the specified user id.

It recursively searches through all descendants of the specified account and returns only those accounts whose email, fullname or guid matches the filter.

The provided user-id must be the authenticated client user or a descendent thereof.

It is possible to limit the depth of search by setting max_depth option:

1 - means that it will only search immediate children, making this call equivalent to calling less "heavy" method GET /users/{user-id}/users without any options. Default value is 1.

Options limit and offset are used to limit the number of items returned in XML response. Default limit is 9999. Default offset is 0;

To sort the results by accounts full-name the option order_by_name should be set to 1. Otherwise the results will be sorted by internal account id and path, i.e. first will be displayed all children of the account with lower id. Default value is 1.

The option reverse_order inverts the sorting. Default value is 0.

The all option is used to get list of all user accounts: active and scheduled for deletion (all=1) or only active users(all=0). Default value is 0.

Response

The response document adheres either the schema v0 (default):

```
element users {
 element id { text }*
}
```

Or to schema v1:

```
element users {
 user {
   element id { text }
   element created { iso8601-datetime }
   element parent_id { text }
   element email { text }
   element full_name { text }
   element deletion-deadline { timestamp }?
 }*
}
```

The selection of the desired response schema is done by setting header Accept.

I.e custom media types are used in the API to let consumers choose the format of the data they wish to receive.

```
Accept: application/vnd.veritas.v{0|1}+xml
```

If Accept header does not have version number then response schema v0 will be used.

If no sub-users have been found matching the filter then HTTP error code 404 Not Found is returned:

```
"No child accounts have been found matching the criteria"
```

If at least one account is found then the response will have the code 200 OK and appropriate XML body.

The number of total sub-accounts found is returned in custom header:

```
x-total-count: {number}
```

The value of x-total-count is not affected by setting limit and offset options.

Example of use:

```
curl -u root@test.com:***** "https://localhost:8501/users/lbqomg-m1x5m0-
lfcall/users?filter=&max_depth=99&limit=100&offset=0&order_by_name=1&reverse_order=0" -
k -H "Accept: application/vnd.veritas.v2+xml" -v
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user-id}/attributes

This method is used list all attributes assigned to a given user account.

Response

```
element attributes {
 element attribute {
  element name { text }
  & element value { text }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method GET /users/{user-id}/attributes/{key}

This method is used to get value for specified key of defined user.

The value is UTF8 string.

Example

```
GET /users/asdf-2345-1233-asdf/attributes/some_token


200 OK
content-type: application/application/octet-stream
content-size: 31

mwV1vtp6wKfhAfkMRFp-1BRdCwl9.cvGF
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method PUT /users/{user-id}/attributes/{key}

This method is used to create/update value for specified key of defined user.

The value is UTF8 string.

Request

```
PUT /users/asdf-2345-1233-asdf/attributes/some_token
content-type: application/octet-stream
content-size: 31


mwV1vtp6wKfhAfkMRFp-1BRdCwl9.cvGF
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method DELETE /users/{user-id}/attributes/{key}

This method is used to delete specified key of defined user.

Example

```
DELETE /users/asdf-2345-1233-asdf/attributes/some_token
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Attributes for GUI theme customization

All theme_root_* fields are relative URLs that points to Veritas shared folders that contain HTML and graphical resources that may be used for building customised look of website and emails.

- **theme_root_inherited** contains URL that was inherited from parent account settings;
- **theme_root_private** contains URL that will be visible only for current account and is supposed to be used for testing purposes during customisation by partner;
- **theme_root_descendants** is a URL that will be propagated to child accounts of a partner and for such child accounts it will be represented as theme_root_inherited field;
- **theme_admin** is a boolean flag that will tell web-client whether to display theme configuration panel (only partners are supposed to edit themes). Currently is will be true for all accounts that have children.

Typical usage scenario:

Default Veritas SaaS Backup website theme will be referred by theme_root_inherited URL for all partner accounts.

If they do not wish to use their custom theme (folder with HTML and graph resources) they should just leave theme_root_descendants empty. It means that their descendant accounts will just use the URL where points theme_root_inherited of this parent account.

If partner wishes to propagate its own UI customization over his clients - he just sets up theme_root_descendants field and it will be visible as field theme_root_inherited for all his children.

Before partner set theme_root_descendants it would make sense to try the theme themselves in order to see how it looks. For this purpose the URL theme_root_private will allow setting up theme only for partner account (w/o propagating further to children) in order to see how it looks. And only when partner is satisfied with look of the chosen theme he stores the URL at theme_root_descendants.

Attributes for SMTP mail delivery choice

A user can explicitly allow or disallow delivery of messages via SMTP to the e-mail address of the primary contact of the account. If SMTP delivery is neither explicitly allowed nor disallowed, the setting is taken from the parent account (and so on recursively until it is set). If no account all the way up to the root sets this, SMTP delivery is considered disallowed.

On the account, the attribute **allow-email** can be set to either **true** or **false**. This will either allow or disallow SMTP delivery of mail to the given user.

Since this attribute is not set by default, any account will use the setting of its parent.

Attributes for product options

Product options have flexible settings. This means if option is not explicitly set for the product it can be set for a particular user.

If attribute is not set option is considered disabled.

The next attributes are available (all boolean):

- publiclinks - allow link sharing
- pcs - allow user create physical devices

- clouds - allow user create cloud devices
- drives - allow user create online drives
- products - allow user create products
- localization - give user access to localization
- messages - give user access to mails
- statistics - give user access to statistics
- themes - allow user edit themes
- groups - allow user create groups

Method GET /users/{user_id}/theme_config

This method is used to get GUI theme configuration for currently logged in user

Response

```
element theme_config {
 element theme_root_inherited { text }
 & element theme_root_private { text }
 & element theme_root_descendants { text }
 & element theme_admin { boolean }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method PUT /users/{user_id}/theme_config

The method allows to update theme settings

Request

```
element theme_config {
 element theme_root_private { text } ?
 & element theme_root_descendants { text } ?
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	FAIL	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user_id}/groups

This method allows retrieval of which groups the user is a member of. Please note, these are not the groups configured by the account, these are the groups the account is a member of.

Response

```
element groups {
 element group {
```

```
 element name { text },
 & element alias { text }?
 & element guid { text }
 & element provisions-enabled { boolean }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user_id}/groups/{group-id}/provisions

Returns list of provisions configured for the group {group-id} user belongs to.

Response

```
element provisions {
 element resource {
   element name { text },
   & element limit { text }? # resource limit. Makes sense for integer resources.
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	OK	FAIL

Method GET /users/{user_id}/groupconfig

This method retrieves all groups configured under the given account. Please note, this is NOT the memberships, these are the accounts configured by the group.

Response

```
element groups {
 element group {
 element name { text },
 & element alias { text }?
 & element guid { text }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method POST /users/{user_id}/groupconfig

Allows creation of a new group, given a name. The location header in the response will contain the full path to the created group.

Request

```
element group {
 element name { text }
 & element alias { text }?
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user_id}/groupconfig/{group_id}

This method retrieves a group configuration

Response

```
element group {
 element name { text }
 & element alias { text }?
 & element autoadd { boolean }
 & element provisions-enabled { boolean }
 & element member-account { guid }*
 & element default-drive { guid }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /users/{user_id}/groupconfig/{group_id}

This method updates a group configuration

Response

```
element group {
 element name { text }?
 & element alias { text }?
 & element autoadd { boolean }?
 & element provisions-enabled { boolean }?
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}

This method deletes a group - returns either status 200 on success or 404 if group cannot be found.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user_id}/groupconfig/{group_id}/provisions

Returns a list of provisions configured for the group. Provisions are a subset of product resources.

Response

```
element provisions {
  element resource {
    element name { text }
    element limit { text }? # resource limit. Makes sense for integer resources.
  }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user_id}/groupconfig/{group_id}/provisions

Adds a new provision (resource) to the list of group provisions. Provision must have the same type as respective product resource.

Request

```
element resource {
  element name { text }
  element limit { text }? # resource limit. Makes sense for integer resources.
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}/provisions/{name}
Removes given provision from the list of group provisions.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user_id}/groupconfig/{group_id}/members/
Adds an account as a member of the group.

Request
```
element member {
 element account { guid }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}/members/{account_id}
Removes an account as a member of the group.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user_id}/groupconfig/{group_id}/drives/
Adds a device as a default-device of the group.

Request
```
element device { guid }
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}/drives/{account_id}
Removes a default-drive from the group.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user_id}/groupconfig/{group_id}/permissions

Adds a permission to the group

Request

```
element permission {
 element user-share {
  element account { guid }
 }
 | element group-share {
    element group { guid }
   }
 | element ip-share {
    element ip { text }
   }
 | element anonymous { }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}/permissions/{perm}

Removes permissions without permission-data ('anonymous' permission only for now) from the group

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/groupconfig/{group_id}/permissions/{perm}/{perm-data}

Removes permissions with permission-data (user-share, group-share, etc.) from the group.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user_id}/access_requests

Creates pending access request to subuser's account.

Request

```
element access {
  element account { guid }
}
```

201 is returned on success, 409 if request has already been placed.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user_id}/access_requests

Returns pending access requests to user's subaccounts.

Response

```
element access_requests {
  element access {
    element guid { guid }
    & element account { guid } # guid of the account request is sent to
  }*
}
```

200 is returned on success.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user_id}/access_requests/pending

Returns pending access requests to user's account. User can grant or reject them.

Response

```
element access_requests {
  element access {
    element guid { guid }
    & element account { guid } # guid of requesting account
  }*
}
```

200 is returned on success.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	FAIL	FAIL	FAIL

Method POST /users/{user_id}/access_requests/{request-id}
Grants pending access request to user's account.

Request
Body is always empty.

Response
Is always empty.

200 is returned on success.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	FAIL	FAIL	FAIL

Method DELETE /users/{user_id}/access_requests/{request-id}
Cancels request if run as administrator or partner. Rejects pending access request if run as a user.

200 is returned on success.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user-id}/storage_island
This request is used only by our internal account to view current and previous (if not empty) storage islands of any specific user.

Response
The response body document adheres to the following schema

```
element osisland {
 element current { text }
 & element previous { text }?
}
```

200 is returned on success.

ACL
To use this endpoint set "StorageIsland" ACL.

Method PUT /users/{user-id}/storage_island
This request is used only by our internal account to update current storage island of any specific user.

Note that previous storage island must be empty to be able to use this endpoint.

Request
The request body document must adhere to the following schema

```
element osisland {
 element current { text }
}
```

200 is returned on success. If success, then previous storage island will be set from the current value, and then, the new value will be assigned to current.

403 can be returned when previous storage island is not empty.

ACL
To use this endpoint set "StorageIsland" ACL.

Method DELETE /users/{user-id}/storage_island
This request is used only by our internal account to clear previous storage island of any specific user.

ACL
To use this endpoint set "StorageIsland" ACL.

ACL API

The ACL (Access Control Layer) or RBAC (Role Based Access Control) system is used to determine whether a given subject is allowed to perform a given operation on a given object.

- The subject is identified by the authentication (authorize header) in the HTTP request.
- The object is the API endpoint.
- The operation is the HTTP method.

Using various mechanisms described on this page, the proxy is able to resolve the effective ACL of a subject (the request initiator) and will use that ACL to determine if the requested HTTP method may be applied to the requested URI endpoint. Any method/endpoint mentioned in the ACL is allowed - any method/endpoint not listed in the ACL is denied.

The ACL string format
An ACL is a text string that is easily parsed by machines and readable by humans. The grammar for the string looks like:

```
ACL := [ entry , { ":" , entry } ]
entry := concrete | reference
concrete := method , { method } , "/" , endpoint
endpoint := epchar , { epchar }
epchar := "A" | ... "Z" | "a" | ... "z" | "0" | ... "9" | "-" (* upper/lower case a-z
plus 0-9 and dash *)
method := "G" | "O" | "D" | "H" | "U" | "P"

reference := epchar , { epchar }
```

ACL entries can be named; the name does not appear in the ACL string itself, but an ACL string can refer to a named ACL which effectively means the named ACL is resolved and included in the referring ACL. Such references can nest to any level and the evaluator must prevent infinite recursion.

Some example ACL strings are listed below – we group the strings either by name or by access token id (for which the string is assigned)

```
User:
GPO/Tokens:UD/Token:G/UserGroups:G/UserProvisions:G/UserGroupConfig:G/Group:G/Search:G/
BSearch:GO/Users:GUD/User:G/Resources:G/UserProduct:GP/Contacts:GU/Contact:GO/SubUsers:
G/UserAttrs:GUDO/UserAttr:G/UITheme:G/AzureStorageConfig:GP/Devices:UD/Device:GPU/Histo
ry:G/HistoryLatest:G/DevStatus:G/DevHealth:P/DevNotify:GPD/DevPeers:D/DevPeerAcct:G/Dev
Attrs:GUD/DevAttr:P/DevAuthCode:GPUDHO/Portfolio:GPH/Object:GPUDHO/BLAPI:P/DevRestoreIt
ems:GH/DownFile:P/Queue:D/QueueEvent:G/Status:G/Time:GP/Favourites:D/Favourite:GP/Share
s:G/UserShares:GUD/Share:G/SPath:G/SFile:G/SFileMetaInfo:G/BPath:PUD/DevCurrent:PO/Rese
tPass:G/VQueue:P/VQEvent:GPU/Perf:GPUDHO/Log:GU/UserLog:GPUDHO/UserCC:GPUDHO/UserInvoic
es:G/CentralConfig:G/DevCentralConfig:G/CentralConfigRules:G/CentralConfigRule:G/Pendin
gAccessRequests:PD/AccessRequest:GP/SsoLogin:G/SsoMetadata

AnonymousParent:
O/Tokens:GO/Users:PO/SubUsers:O/UserAttr:GPUDHO/Portfolio:GPUDHO/BLAPI:GH/DownFile:G/St
atus:G/Time:G/SPath:G/SFile:G/SFileMetaInfo:PO/ResetPass:GPU/Perf:GPUDHO/Log:GPUDHO/Use
rCC:GPUDHO/UserInvoices:GP/SsoLogin:G/SsoMetadata

42:
User

44:
User:GO/Time:GO/Status

45:
GO/Time:AnonymousParent
```

If we evaluate the ACLs for the given examples here we will compute the effective ACLs to be

```
42:
GPO/Tokens:UD/Token:G/UserGroups:G/UserProvisions:G/UserGroupConfig:G/Group:G/Search:G/
BSearch:GO/Users:GUD/User:G/Resources:G/UserProduct:GP/Contacts:GU/Contact:GO/SubUsers:
G/UserAttrs:GUDO/UserAttr:G/UITheme:G/AzureStorageConfig:GP/Devices:UD/Device:GPU/Histo
ry:G/HistoryLatest:G/DevStatus:G/DevHealth:P/DevNotify:GPD/DevPeers:D/DevPeerAcct:G/Dev
Attrs:GUD/DevAttr:P/DevAuthCode:GPUDHO/Portfolio:GPH/Object:GPUDHO/BLAPI:P/DevRestoreIt
ems:GH/DownFile:P/Queue:D/QueueEvent:G/Status:G/Time:GP/Favourites:D/Favourite:GP/Share
s:G/UserShares:GUD/Share:G/SPath:G/SFile:G/SFileMetaInfo:G/BPath:PUD/DevCurrent:PO/Rese
tPass:G/VQueue:P/VQEvent:GPU/Perf:GPUDHO/Log:GU/UserLog:GPUDHO/UserCC:GPUDHO/UserInvoic
es:G/CentralConfig:G/DevCentralConfig:G/CentralConfigRules:G/CentralConfigRule:G/Pendin
gAccessRequests:PD/AccessRequest:GP/SsoLogin:G/SsoMetadata

44:
GPO/Tokens:UD/Token:G/UserGroups:G/UserProvisions:G/UserGroupConfig:G/Group:G/Search:G/
BSearch:GO/Users:GUD/User:G/Resources:G/UserProduct:GP/Contacts:GU/Contact:GO/SubUsers:
G/UserAttrs:GUDO/UserAttr:G/UITheme:G/AzureStorageConfig:GP/Devices:UD/Device:GPU/Histo
ry:G/HistoryLatest:G/DevStatus:G/DevHealth:P/DevNotify:GPD/DevPeers:D/DevPeerAcct:G/Dev
Attrs:GUD/DevAttr:P/DevAuthCode:GPUDHO/Portfolio:GPH/Object:GPUDHO/BLAPI:P/DevRestoreIt
ems:GH/DownFile:P/Queue:D/QueueEvent:GO/Status:GO/Time:GP/Favourites:D/Favourite:GP/Sha
```

```
res:G/UserShares:GUD/Share:G/SPath:G/SFile:G/SFileMetaInfo:G/BPath:PUD/DevCurrent:PO/Re
setPass:G/VQueue:P/VQEvent:GPU/Perf:GPUDHO/Log:GU/UserLog:GPUDHO/UserCC:GPUDHO/UserInvo
ices:G/CentralConfig:G/DevCentralConfig:G/CentralConfigRules:G/CentralConfigRule:G/Pend
ingAccessRequests:PD/AccessRequest:GP/SsoLogin:G/SsoMetadata

45:
O/Tokens:GO/Users:PO/SubUsers:O/UserAttr:GPUDHO/Portfolio:GPUDHO/BLAPI:GH/DownFile:G/St
atus:GO/Time:G/SPath:G/SFile:G/SFileMetaInfo:PO/ResetPass:GPU/Perf:GPUDHO/Log:GPUDHO/Us
erCC:GPUDHO/UserInvoices:GP/SsoLogin:G/SsoMetadata
```

As can be seen, the effective ACL is the union of all (method x endpoint) tuples referred in the ACL and all referenced ACLs.

A note about parsing

All entries are separated by semicolons which are invalid anywhere else in the grammar. This makes for easy entry separation.

All concrete entries must contain a slash character, and all reference entries cannot contain a slash character - this also allows easy separation of the two.

A simple graph coloring scheme will allow for simple and efficient prevention of infinite recursion during evaluation of ACL entries that refer to one another.

Named ACLs and their hierarchy

Each account can define named ACLs to be referenced by access tokens created under that account or under any descendent account (direct or indirect child accounts).

The system also defines a few named ACLs; these are stored in the database with "no account"; a NULL as the owning account.

As of this writing, the "system" ACLs are set up to match our previous AT_User, AT_PartnerParent, AT_... token types and are named as "User", "PartnerParent" and so forth. Thus, when transitioning from the old hard-coded ACL system with fixed token types to the new flexible system, any AT_User type token will simply have an ACL string of "User" (in other words, it will reference the system ACL named "User").

No ACL may be created that has the name of one of the system ACLs.

When resolving an ACL reference, a search is done first among the system ACLs, then from the current account and upwards in the tree towards the root. The first matching ACL will be merged as the referenced ACL.

One account may define a named ACL that is already defined under a parent or child account (directly or indirectly).

A note on risk by using named ACLs

We really must allow accounts to define ACLs with names that are already used in either parent or child accounts – otherwise the naming of ACLs will become much too restrictive (and we will leak information about names already present elsewhere in the tree).

Obviously, this introduces a problem upon deletion of an ACL – consider the following account tree with named ACLs:

```
root
|
+-- account-A  (ACL "user" defined as GD:/Users)
|   |
|   +-- account-B (ACL "user" also defined but as G:/Users)
|       |
|       +-- account-C (ACL set to "user")
```

Initially, the ACL of account-C resolves to "G:/Users".

Now, if account-B decides to delete the named ACL "user", this changes the effective ACL of account-C to "GD:/Users". Suddenly a user is granted an extra permission because the parent account deleted an ACL.

Supported ACLs

All ACLs are divided in 2 categories: URI sentinels and features.

Each URI sentinel protects a single endpoint supported by the API server. Features provide access to advanced functionality i.e. allow to impersonate other users or modify arbitrary tokens in the system.

URI sentinels:

- User API
  - **Tokens**
    - G/Tokens - Get all tokens created for the account
    - P/Tokens - Create a new token
    - D/Tokens - Delete all secondary tokens of authenticated user
  - **Token**
    - U/Token - Update token: change user password, login, description, lifetime, ACLs
    - D/Token - Delete token
  - **TokenContacts**
    - G/TokenContacts - retrieve token's contacts
    - U/TokenContacts - update or add token's contacts
  - **ResetPass**
    - P/ResetPass - submit reset password request
- Account API
  - **Users**
    - G/Users - Get guid of authenticated account
  - **User**
    - G/User - Get account information by guid
    - U/User - Update account information: change product, enable/disable, set grace time, subscribe/unsubscribe, change parent account
    - D/User - Delete account by guid
  - **UserProduct**
    - G/UserProduct - Get information about effective user product
  - **UserProductProps**
    - G/UserProductProps - Get properties of the product currently assigned to the account

        - U/UserProductProps - Update/set property of the product currently assigned to the account
        - D/UserProductProps - Delete property/properties of the product currently assigned to the account
    - **Contacts**
        - G/Contacts - Get available types of contact information for the account (p or s)
        - P/Contacts - Create a new contact record for the account
    - **Contact**
        - G/Contact - Get available contact information by type
        - U/Contact - Update contact details
    - **SubUsers**
        - G/SubUsers - Get list of subaccount guids
        - P/SubUsers - Create a new subaccount
    - **UserAttrs**
        - G/UserAttrs - Get all account attributes
    - **UserAttr**
        - G/UserAttr - Get account attribute by name
        - U/UserAttr - Update or create account attribute
        - D/UserAttr - Delete account attribute
    - **Resources**
        - G/Resources - Get resources available to the account
    - **MoveUsers**
    - **UserRevive**
        - U/UserRevive - Recover account scheduled for the deletion
    - **AccessRequests**
        - G/AccessRequests - Get list of access requests to other accounts submitted by the account
        - P/AccessRequests - Submit access request to another account
    - **PendingAccessRequests**
        - G/PendingAccessRequests - Get list of accounts requesting access to the account
    - **AccessRequest**
        - P/AccessRequest - Grant account access request
        - D/AccessRequest - Reject account access request
    - **FCMRecipients**
        - G/FCMRecipients - Get all recipients
        - P/FCMRecipients - Create new recipient
        - U/FCMRecipients - Update recipient details
        - D/FCMRecipients - Delete recipient
- Invoicing and payments API
    - **UserCC**

        - GPUDHO/UserCC - Access to account credit card information
    - **UserInvoices**
        - GPUDHO/UserInvoices - Access to account invooices
- Storage backend API
    - **AzureStorageConfig**
        - G/AzureStorageConfig - Get config of the azure storage backend
        - U/AzureStorageConfig - Update config of the azure storage backend
    - **StorageIslands**
        - G/StorageIslands - Get list of available storage islands in the system. Must not be available to regular users.
    - **StorageIsland**
        - G/StorageIsland - Get storage island configuration of the account. Must not be available to regular users.
        - U/StorageIsland - Set storage island for the account. Must not be available to regular users.
        - D/StorageIsland - Clear previous storage island for the account. Must not be available to regular users.
- Device API
    - **Devices**
        - G/Devices - Get list of account devices
        - P/Devices - Create a new device
    - **Device**
        - G/Device - Get device information
        - U/Device - Update device
        - D/Device - Delete device
    - **EnumerateDevices**
        - U/EnumerateDevices - Get system-wide list of devices marching search criteria. Must not be available to regular users.
    - **History**
        - GU/History - Get list of snapshots
        - P/History - Create a new snapshot
    - **HistoryLatest** - deprecated
    - **DevStatus**
        - G/DevStatus - Get device status
        - P/DevStatus - Update device status
    - **DevJobs**
        - G/DevJobs - Get list of jobs
        - U/DevJobs - Get list of jobs applying filters
        - P/DevJobs - Create job
    - **DevJobsUpdate**

    - P/DevJobsUpdate - Update device jobs
  - **DevJob**
    - G/DevJob - Get job
    - U/DevJob - Update job
  - **DevJobStats**
    - G/DevJobStats - Get job statistics
    - U/DevJobStats - Update or create job statistic
    - D/DevJobStats - Delete job statistic
  - **DevHealth**
    - G/DevHealth - Get device health
  - **DevNotify**
    - P/DevNotify - Send FCM notification to the device. Only for internal use.
  - **DevPeers**
    - G/DevPeers - Get list of accounts having access to shared device (online drive)
    - P/DevPeers - Add a new peer to shared device
    - D/DevPeers - Stop accessing shared device
  - **DevPeerAcct**
    - D/DevPeerAcct - Revoke access to shared device for the account specified by guid
  - **DevAttrs**
    - G/DevAttrs - Get all device attributes
  - **DevAttr**
    - G/DevAttr - Get device attribute by name
    - U/DevAttr - Update or create device attribute
    - D/DevAttr - Delete device attribute
  - **DevRevive**
    - U/DevRevive - Recover device scheduled for deletion
  - **DevHealthCheck**
    - P/DevHealthCheck - Trigger device health check. Must be used only by system accounts and never exposed to end users.
  - **DevResourceUsage**
    - U/DevResourceUsage - Report resource consumption of the device. Must be used only by system accounts and never exposed to end users.
  - **DevResources**
    - G/DevResources - Get the resource consumption reported by a device.
- File downloading API
  - **Object**
    - G/Object - Get object by hash
    - P/Object - Create object in object store

    - H/Object - Check if object exists in object store
  - **DownFile**
    - G/DownFile - Download file
    - H/DownFile - Check if file exists
  - **BPath**
    - G/BPath - Access file by snapshot hash and path. Must be used only by system accounts and never exposed to end users, because it requires snapshot hash which is a sensitive information.
- Online drives API
  - **DevCurrent**
    - P/DevCurrent - Upload file to online drive
    - U/DevCurrent - Rename/move file in online drive
    - D/DevCurrent - Delete file from online drive
- Central config API
  - **CentralConfigs**
    - G/CentralConfigs - Get list of central configurations
    - P/CentralConfigs - Create a new central configuration
  - **CentralConfig**
    - G/CentralConfig - Get central configuration by guid
    - U/CentralConfig - Modify central configuration
    - D/CentralConfig - Delete central configuration
  - **DevCentralConfig**
    - G/DevCentralConfig - Get central configuration applied to the device
    - P/DevCentralConfig - Assign central configuration to the device
    - D/DevCentralConfig - Remove central configuration from the device
  - **CentralConfigDevices**
    - G/CentralConfigDevices - Get list of devices central configuration is assigned to
  - **CentralConfigRules**
    - G/CentralConfigRules - Get rules defined by central configuration
    - P/CentralConfigRules - Add a new rule to the configuration
  - **CentralConfigRule**
    - D/CentralConfigRule - Delete a rule from central configuration
- Group API
  - **UserGroups**
    - G/UserGroups - Get list of groups account belongs to
  - **UserProvisions**
    - G/UserProvisions - Get list of provisions configured for the group user is member of
  - **UserGroupConfig**
    - G/UserGroupConfig - Get list of groups created by the account

        - P/UserGroupConfig - Create a new group under the account
    - **Group**
        - G/Group - Get information about a group including members
        - U/Group - Modify group information
        - D/Group - Delete group
    - **GroupMembers**
        - P/GroupMembers - Add an account to the group
    - **GroupMember**
        - D/GroupMember - Remove an account from a group
    - **GroupDrives**
        - P/GroupDrives - Create group drive
    - **GroupDrive**
        - D/GroupDrive - Delete group drive
    - **GroupProvisions**
        - G/GroupProvisions - Get the list of provisions configured for a group
        - P/GroupProvisions - Add provision to the list of provisions of a group
    - **GroupProvision**
        - D/GroupProvision - Remove provision from the list of provisions of a group
    - **GroupPerms**
        - P/GroupPerms - Add a new rule to the list of sharing rules of the group
    - **GroupPermData**
        - D/GroupPermData - Remove a rule from the list of sharing rules of the group
- Search API
    - **Search**
        - G/Search - deprecated?
    - **BSearch**
        - G/BSearch - Perfom search queries
    - **BData**
        - P/BData - Update searchable dataset. Must be used only by system accounts and never exposed to end users.
        - D/BData - Delete searchable dataset. Must be used only by system accounts and never exposed to end users.
- Cloud connectors API
    - **DevAuthCode**
        - P/DevAuthCode - Exchange OAuth auth code to access and refresh tokens
    - **O365** - deprecated
    - **O365ClientId**
        - G/O365ClientId - Get id of the Office365 application we use to backup user's account

    - **O365SiteCollections**
        - G/O365SiteCollections - Get list of the Office365 site collections
    - **O365Sites**
        - G/O365Sites - Get list of the Office365 top level sites
    - **O365SubSites**
        - G/O365SubSites - Get list of the Office365 site subsites
    - **O365Users**
        - G/O365Users - Get list of the Office365 users
    - **O365Groups**
        - G/O365Groups - Get list of the Office365 groups
    - **O365ExpGroups**
        - G/O365ExpGroups - Get expanded list of the Office365 groups, including their members
    - **O365GroupMembers**
        - G/O365GroupMembers - Get list of members of the Office365 groups
    - **Dynamics365ClientId**
        - G/Dynamics365ClientId - Get id of the Dynamics365 application we use to backup user's account
    - **GSuiteUsers**
        - G/GSuiteUsers - Get list of the GSuite users
    - **GSuiteDomains**
        - G/GSuiteDomains - Get list of the GSuite domains
    - **GSuiteClientId**
        - G/GSuiteClientId - Get id of the the GSuite application we use to backup user's account
    - **SalesforceClientId**
        - G/SalesforceClientId - Get id of the the Salesforce application we use to backup user's account
    - **SalesforceUsersCount**
        - G/SalesforceUsersCount - Get list of the Salesforce users
- Favourites API
    - **Favourites**
        - P/Favourites - Create a new favourite
        - G/Favourites - Get list of favourites
    - **Favourite**
        - D/Favourite - Delete favourite
- Sharing API
    - **Shares**
        - P/Shares - Create a new share
        - G/Shares - Get list of shares
    - **UserShares**

        - G/UserShares - Get list of shares as above but via another endpoint. Will be deprecated and replaced with 'G/Shares' ACL, but currenly must be specified together with G/Shares
    - **Share**
        - G/Share - Access share page
        - U/Share - Update share
        - D/Share - Delete share
    - **SPath**
        - G/SPath - Browse shared folder
    - **SFile**
        - G/SFile - Access shared file
    - **SFileMetaInfo**
        - G/SFileMetaInfo - Get information about shared file
- Product portfolio API
    - **Portfolio**
        - GPUDHO/Portfolio - Access to product portfolio configuration
    - **Product** - deprecated?
    - **ProductUsers** - deprecated?
- Single sign-on API
    - **SsoLogin**
        - GP/SsoLogin - Access to SSO login endpoints. Must be enabled in order to allow to login via SSO.
    - **SsoMetadata**
        - G/SsoMetadata - Return SSO metadata. Must be enabled in order to allow to login via SSO.
    - **SsoConfigs**
        - G/SsoConfigs - Get list of SSO configurations
        - P/SsoConfigs - Create a new SSO configuration
    - **SsoConfig**
        - G/SsoConfig - Get information about SSO configuration
        - U/SsoConfig - Update SSO configuration
        - D/SsoConfig - Delete SSO configuration
- Log API
    - **Log**
        - GPUDHO/Log - Access to log endpoints
    - **UserLog**
        - GU/UserLog - Query logs as a user. Requires **G/LogFilter** ACL
- Job API
    - **NextJob**

        - G/NextJob - Get next job scheduled for execution by cloud connector scheduler. Must be used only by system accounts and never exposed to end users.
    - **JobsOfState**
        - G/JobsOfState - Get list of jobs by state: scheduled, cancelled, in progress, succeeded, etc
- Queueing API
    - **Queue**
        - G/Queue - Get next event from event queue
        - P/Queue - Add new event to the queue
    - **QueueEvent**
        - D/QueueEvent - Delete event from the queue
    - **VQueue**
        - G/VQueue - Subscribe to volatile message queue events
    - **VQEvent**
        - P/VQEvent - Post message to volatile event message queue
- Misc API
    - **BLAPI**
        - GPUDHO/BLAPI - Generic ACL enabling access to multiple endpoints in business logic. Must be deprecated and replaced with more specific ACLs
    - **UITheme**
        - G/UITheme - Get custom UI theme settings
        - U/UITheme - Set custom UI theme setting
    - **Status**
        - G/Status - Get server status
    - **Time**
        - G/Time - Get server time
    - **Perf**
        - GPU/Perf - Access to performance metering system
    - **GlobalAttrs**
        - G/GlobalAttrs - Get global attributes
        - U/GlobalAttrs - Update/create global attributes
        - D/GlobalAttrs - Delete global attributes
    - **BackupRetentionConfig**
        - G/BackupRetentionConfig - Get cached backup retention config of an account. Must be used only by system accounts and never exposed to end users.
        - U/BackupRetentionConfig - Cache backup retention config for an account. Must be used only by system accounts and never exposed to end users.

Features

- **G/CreatePrimaryToken** - Allow to create primary tokens
- **G/ChangeSiblingTokens** - Allow token to change all others tokens of account, including sibling tokens

- **G/MaySetPrimaryPassword** - Allow secondary tokens to change password of their primary token. Should never be explicitly given to the user.
- **G/MaySetParentToken** - Allow to set any parent token via "x-set-parent-token" header when creating secondary tokens. Must be used only by system accounts and never exposed to end users.
- **G/Impersonate** - Allow to impersonate any account in the system. Must be used only by system accounts and never exposed to end users.
- **G/ForestAccess** - Allow accessing accounts we're not the ancestor of. By default account has access only to own subaccounts. Must be used only by system accounts and never exposed to end users
- **G/GraceExpirySet** - Allow setting grace expiry on an account. Must be used only by system accounts and never exposed to end users.
- **G/AccountMove** - Allow setting the parent on an account
- **G/HardShareAccess** - Allow access to password protected shares without providing a password. Usually shouldn't be available to end users.
- **G/CreateBaseUser** - Allow creation of user account with access token with "basic" access rights ("User")
- **G/CreateAnyAclToken** - Allow to create tokens with any ACL set. By default token can only create tokens with more narrow ACL set than it has. Must be used only by system accounts and never exposed to end users.
- **G/CloudCredentials** - Some legacy connectors i.e. ftp use login/password to access the data. By default this sensitive information is not returned to the user, but may be if this ACL is enabled. Usually shouldn't be available to end users.
- **G/ModifySysAttr** - Allows to modify system-tagged ("s-" prefixed) attributes which are read-only by default. Must be used only by system accounts and never exposed to end users.
- **G/ForeignHashes** - Snapshot hashes are only returned to the owner (or peer if device is shared) of the device. This ACL allows seeing snapshot hashes of other people histories. Must be used only by system accounts and never exposed to end users.
- **G/WildcardMQ** - Allow subscribing to wildcard message queues (queue-*). Usually shouldn't be available to end users.
- **G/MessageAPI** - Allow message API access in buslog
- **G/TrustedLogger** - Accept log entries on behalf of arbitrary accounts/devices. Must be used only by system accounts and never exposed to end users.
- **G/LogFilter** - Allow searching/filtering on logs
- **G/FilteredEventMQ** - Allow subscribing to filtered event log data
- **G/UnfilteredLogData** - Allow view original messages on logs (e.g. messages with hashes) Must only be given to trusted or internal users.
- **GU/AuditFilter** - Allow searching/filtering on audit logs
- **G/InternalAudit** - Allow searching/filtering on internal audit logs. Usually shouldn't be available to end users.
- **P/CCScheduleBackup** - Schedule backup jobs
- **P/CCScheduleRestore** - Schedule restore jobs
- **P/CCScheduleSRestore** - Schedule single-file restore jobs
- **P/CCRestoreDirModeDeltaAppend** - create new files and to overwrite existing files(if they differ in the snapshot and in the target service) during restore
- **P/CCRestoreDirModeDeltaRestore** - same as DeltaAppend but also allows to remove files that are present in the target service but are not present in the snapshot

- **P/CCRestoreDirModeWipeAndRestore** - not used
- **P/CCRestoreFileModeRestore** - if the file in the snapshot differs from the file in the target service - overwrite the file in the target service
- **P/CCRestoreFileModeRename** - if the file in the snapshot differs from the file in the target service - create a new file in the service with a new name, leave the existing file as is
- **P/CCRestoreFileModeSkip** - if the file in the snapshot differs from the file in the target service - do nothing
- **P/CCRestoreMethodInPlace** - Restore data into the same place it was backed up from
- **P/CCRestoreMethodForceInPlace** - not used
- **P/CCRestoreMethodToFolder** - Create a new folder in the target service and restore data into it, so that the current data is not affected/overwritten
- **G/Captchaless** - do not require to solve captcha when creating a new user
- **AttemptAuth** - This feature is used to store failed authentication attempts in audit log. This is rather dummy ACL and shouldn't be explicitly given to any user.
- **G/Reports** - Currently, this is dummy feature and it's not used. Later, will be used to control reports
- **P/MayCreatePmrJobs** - Allow creation of PMR jobs
- **P/MayCreatePstRestoreJobs** - Allow downloading of data in PST format

Authentication services API

The user database provides the following services:

- Authentication services (authenticate user, change password, create access token, revoke access token)
- Account services (provisioning, deletion)
- Data access services (backup history, archive, read-only-share)

Location services - data must be able to reside in local data centers. We need proxy names like my.saasbackup.veritas.com for user accounts.

Message encoding

All structured request and reply body data in the UD/API will be XML. Each request will specify the schema using RNC.

Authentication services

In general, requests that require authentication can be authenticated simply by providing the username and password using Basic HTTP authentication. Since the publicly available protocol is provided only over HTTPS the username/password combination is never transmitted in clear text (except internally among our back-end systems).

The authentication services can be used to create "access tokens" - for use with mobile and native PC clients, so that these do not need to store the user credentials locally (and so that changing the user password on his account does not require password updating on all the users clients as well).

All tokens consist of a GUID and a large random string, so they can still be used in the HTTP Basic authentication scheme.

All requests against the UD/API must be authenticated (either by means of an access token or a real username/password).

Method POST /tokens/

Deprecated - please do not use /tokens/ directly anymore. This endpoint has been moved down to under /users/{guid}/tokens.

Method POST /users/{guid}/tokens/

Creates a new temporary or permanently valid access token. The body of the request must adhere to this schema:

POST request body schema

```
element token {
  ( element type { text } # LEGACY, use acl instead - AnonymousParent, PartnerParent,
Administrator, User, Device, System; same as issuer if not supplied
    | element acl { text } )?
  & element descr { text }  # Short textual description of token (eg. name of device)
  & element aname { text }? # Optional authentication-name; GUID will be generated if
not supplied
  & element apass { text }  # Password string
  & (element expires { timestamp } # Date when token would be expired
    | element lifetime { iso8601-period })? # lifetime of token - infinite if not
supplied
  & element device { text }? # GUID of device this token is stored on
  & element primary { boolean }? # true if need to create a primary token
  & element singleuse { boolean }? # true if token will expire after the first usage
 }
```

The URI to the created token is returned in the locationid header of a successful 201 response.

The "type" field specifies the legacy type of token that is being created. If a type is specified, it is translated into an ACL holding a single reference entry referencing the ACL named after the token type. In other words, supplying type "User" will result in an ACL of "User", which can successfully be resolved since the system defines a named ACL with exactly that name.

If neither ACL nor type is provided, the ACL of the currently authenticated access token is used exactly as it is defined there; meaning, references are not first resolved.

It is not possible to create a token with an ACL containing access entries that are not present in the currently effective ACL.

When someone creates a new trial account, a token is created for that account. This token must be of the User type.

When a device connects to the back-end, it will ask the user for credentials - those credentials (which is of type User) are then used to create a Device type access token for the device. The device type token is stored in the device configuration and is used by the device for backups and data access (so that the User credentials are never stored on the device).

Lifetime will set a start time and a period, each time the token is used, this period start time will be set to the current time - "renewing" the token lifetime.

Expires parameter should be set in the iso8601-timestamp format of UTC. If expires param is set then lifetime will be calculated from current time to expires time.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /tokens/

Deprecated and moved to /users/{guid}/tokens.

Method GET /users/{guid}/tokens/

Enumerates all tokens registered on the account.

Response body schema

```
element tokens {
 element token {
  element descr { text }
  & element aname { text }
  & element created { timestamp }   # Time of creation
  & primary primary { boolean } # true for primary tokens
  & element lifetime { timestamp }? # Lifetime, if set. Absense of element means
lifetime is indefinite
  & element acl { text } # Defined ACL of the token in question
  & element eacl { text } # Effective (concrete) ACL of the token in question
  & element primary_aname { text } # Primary token aname
  & element device { text }? # GUID of device this token is stored on
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /tokens/{aname}

Deprecated and moved to /users/{guid}/tokens/{aname}

Method PUT /users/{guid}/tokens/{aname}

To change the name, password or lifetime of a token, an authenticated user can issue this request on any token that authenticates to his account.

It is not possible, currently, to change the lifetime from a finite value to indefinite; when tokens are created, their lifetime is indefinite if a lifetime is not supplied, but in the update message the absence of a lifetime simply means that it is not altered.

```
element token_update {
 element aname { text }?    # Optional name - for renaming of tokens
 & element apass { text }?  # Optional password - for changing password
 & element descr { text }?  # Token description. Can not be empty.
 & (element expires { timestamp } # Optional expires timestamp
   | element lifetime { iso8601-period })? # Optional lifetime
 & (element type { text } # LEGACY, use acl instead - AnonymousParent, PartnerParent,
Administrator, User, Device, System; same as issuer if not supplied
   | element acl { text } )?
 }
```

The URI to the updated token is returned in the locationid header of a successful 201 response.

In order to update a token, the authenticating token (the token passed in the HTTP Basic Authentication header) must be a primary token. This is to prevent a hijacked user session (which uses a non-primary token) from changing the password of the users account (by modifying the primary token).

ACL

Endpoint is named "Token"

Setting of ACL or type requires that the effective ACL of the request initiator dominates the ACL requested.

A primary token can only update itself and its child tokens, unless it has the ChangeSiblingTokens feature in which case it may also modify tokens in sibling trees.

Method DELETE /tokens/{aname}

Deprecated and moved to /users/{guid}/tokens/{aname}

Method DELETE /users/{guid}/tokens/

Deletes all secondary tokens of authenticated token. If ChangeSiblingTokens ACL is present and "x-token-name" header is set to some token name then all secondary tokens of this token would be deleted. We should use this endpoint for logout everywhere feature.

Note: In order to prevent deletion of primary tokens it should never has set up lifetime.

On deletion success the code 200 OK is returned.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method DELETE /users/{guid}/tokens/{aname}

Deletes token of the same type as the type of the token of an authenticated user. In order to prevent deletion of primary tokens. i.e. the ones that contain user's real login/password it is only allowed to delete secondary tokens (normally created by primary token, with auto generated login/pass).

On deletion success the code 200 OK is returned.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Contact information

The token can have its own contact information, but only if it's primary token, secondary token cannot have contacts. By default, a token can change only himself contact information, and only if TokenContacts ACL is set. If ChangeSiblingTokens feature ACL is additionally set than token also can change other primary tokens contacts in the authenticated account.

Method GET /users/{guid}/tokens/{aname}/contacts

Retrieve token's contacts details. G/TokenContacts ACL should be set.

Response body schema

```
element contact {
 & element email { text }?
 & element companyname { text }?
 & element fullname { text }?
 & element phone { text }?
 & element street1 { text }?
 & element street2 { text }?
 & element city { text }?
 & element zip { text }?
 & element state { text }?
 & element country { text }?
 & element language { text }?
}
```

Method PUT /users/{guid}/tokens/{aname}/contacts

Update or add token's contact details. U/TokenContacts ACL should be set.

Request body schema

```
element contact {
 & element email { text }?
 & element companyname { text }?
 & element fullname { text }?
 & element phone { text }?
 & element street1 { text }?
 & element street2 { text }?
 & element city { text }?
 & element zip { text }?
 & element state { text }?
 & element country { text }?
 & element language { text }?
}
```

Account activation

Sometimes it's necessary to send account activation mail manually. The next endpoint serves this purpose. Mail is sent to token's email that is set in his contacts. If no email set, we use token's aname.

Method POST /users/{guid}/activate

P/UserActivate ACL must be enabled for the account.

This request generates temporary single use token with 6 weeks lifetime. Please be sure welcome/service-period-started/trial-period-started (depending on the product configuration) message template contains resetpass tag

```
element activate {
  element token { text }
}
```

201 code is returned in case of successful request.

Reset Password

"Reset password" sequence (implementation details of 'reset password' functionality)

If user forgets their password, but remembers the email they used for registration, they can send a request to Veritas support to reset the password (more precisely, to set new password). Veritas support in turn will send email to the user with a temporary, auto-generated link and instructions about how to change password.

The request should be authenticated by AnonymousParent or PartnerParent credentials.

Method POST /reset_pass/

```
element reset_pass {
  element email { text }
}
```

When request is created the code 201 is returned.

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	FAIL	FAIL	FAIL	FAIL	FAIL

BSearch API v2.0

Intro

BSearch API v2.0 does NOT filter by default. All entries are returned.

Specifying searchTerms or filtering parameters will cut off results that don't meet required conditions.

This means now **deleted and system files WILL be returned by default**.

Use filters to narrow down the results.

If the searchTerms are specified sortTerms are not supported - the default sorting order by relevance is applied.

The standard supported BSearch flows are(other combinations are supported too, these are just the most common):

- Browsing a specific folder - use device, pathRoot, filterAnd/filterOr and sortTerms
- Search the whole device - use searchTerms, device, filterAnd/filterOr and sortTerms
- Search the folder recursively/not recursively - use searchTerms, device, filterAnd/filterOr, sortTerms, pathRoot and recursive

- Query the full history for the file/folder - device, pathRoot, searchTerms="exactfilename" and fullHistory. If you do not want to see file deletions - filter them out using filterAnd/filterOr
- Query the number of versions of file/folder - device, pathRoot, searchTerms="exactfilename", fullHistory, includeBody=0. If you do not want to see file/folder deletions - filter them out using filterAnd/filterOr

BSearch formal grammar:

```
PARAMETER = "pathRoot" | "device" | "fullHistory" | "includeBody" | "recursive"|
"searchTerms" | "sortTerms" | "filterAnd" | "filterOr" | "count" | "startIndex" |
"snaptime" | "snaptimeFrom" | "snaptimeTo" | "apiVersion";
```

VALUE is string or integer depending on the parameter.

Usage:

```
/bsearch?[PARAMETER=VALUE]&[PARAMETER=VALUE]&[PARAMETER=VALUE]...
```

Parameters Explanation

Specifying where to search

Parameter	Explanation	Value
device	If pathRoot was supplied, a device guid must also be specified (to give the path its necessary context).	Valid device guid string.
pathRoot	Full path to the container of files we wish to consider in the search (if not provided, search will search across all objects regardless of location). At least pathRoot or searchTerms must be specified.	Valid path root string.
recursive	Enable recursive search in specified pathRoot. Should only be used when the pathRoot is also provided. The search will be performed in the pathRoot and all it's subfolders.	0; 1; (default 0)

Specifying snapshot and pagination

Parameter	Explanation	Value
count	Maximum number of results in response. If not given, 100 is used.	Any integer.
startIndex	Index of first result. If not given, 0 is used.	Any integer.
snaptime	Disregard any information that appeared after this ISO8601 timestamp (in other words, search/browse device(s) as it/they were at the given time).	Valid ISO8601 timestamp.
snaptimeFrom	Define snapshot range (timestamp from).	Valid ISO8601 timestamp.
snaptimeTo	Define snapshot range (timestamp to).	Valid ISO8601 timestamp.

Specifying what to return

Parameter	Explanation	Value
includeBody	Return full response (default is true). Setting this to false is used to get number of items that satisfy search criteria	0; 1; (default 1)
fullHistory	Should only be used when querying the history of the single file - device, pathRoot and searchTerms="exactFileNameInThePathRoot"	0; 1; (default 0)

Filtering and sorting

Parameter	Explanation	Value
searchTerms	Specify what terms to search for (if not given, all objects will match). This is fuzzy search, support of the exact search has been terminated. Search is case insensitive.	String that is space(or '-', '_', '.', ':') separated list of terms.
itemName	Exact name of the item to search for. It should not be used for classical search but rather for such cases as retrieving history of changes for a given item.	String
sortTerms	Specify sorting criteria.	String that is comma separated list of predefined sorting terms.
filterAnd filterOr	Specify which relation will be between filters. Only one of these parameters can be used at a time.	String containing filters.

Version

Parameter	Explanation	Value
apiVersion	Specify which API version should be used.	1; 2; (default 1)

searchTerms:

```
searchTerms=[TERM] [TERM] [TERM] ...
```

Example:

```
    searchTerms=brown fox
```

sortTerms:

```
sortTerms=[TERM],[TERM],[TERM]...
```

Term	Explanation
title	Sort by value of meta key *subject* if exists, otherwise by value of meta key *dname*. If neither of two exist - by item filename.
name	Sort by item name (aka filename).
updated	Sort by time of snapshot in which item was last updated.
time	Sort by extracted timestamp from value of meta key *date* if present, otherwise from value of meta key *mtimes* if present, otherwise sort by time of snapshot in which item was updated.
key	Lexicographical sort by value of custom meta key.
error	Sort by relevane, that is items with the closest match will be at the beginning. This is default sorting in search scenario, when searchTerms are specified and sortTerms aren't.
audio	Audio files will be at the beginning, others will follow.
video	Video files will be at the beginning, others will follow.
image	Images will be at the beginning, others will follow.
message	Messages will be at the beginning, others will follow.
document	Documents will be at the beginning, others will follow.
folder	Folders will be at the beginning, others will follow.
size	Small size will be at the beginning, others will follow.

Each of these terms can be prefixed with "-", which will cause the sorting order to be reversed. Sorting by string (with terms title, name and custom meta key) is case sensitive.

Example:

```
      sortTerms=title,updated,-name
```

filterAnd and filterOr:

Filter query is written in form similar to Polish notation. It consists of groups which in turn consist of criteria. Firstly, one should specify how groups will be related each with other: filterAnd means that logical AND will connect groups, filterOr will connect groups with logical OR.

Then, groups themselves are key:value pairs, where key could be either AND or OR and values are comma separated list of predefined keywords which in turn are "concatenated" in a way that key specifies. This means that now we have two types of groups: AND-groups and OR-groups.

There is no limit on number of groups that can be specified. Groups are separated by semicolon. Order of the groups is not important, one can start with AND-group as well as OR-group.

```
fillterAnd | filterOr=[AND | OR:[CRITERION],[CRITERION],...];[AND |
OR:[CRITERION],[CRITEREION],...];...
```

Criterion	Explanation	Value
deleted	Should deleted files be returned.	-
key	Should items that contain custom meta key be returned.	"dname" "subject" "sys" "failed" "ext" "exto" "class" "mtimes" "path" "size" "inaccessible" "active" "mail" "upn" "displayName" "id" "redirect" "folder_type" "udt" "contentId". However, this is **NOT** a full list of keys.
key=value	Should items with specified value of some meta be returned.	Any string.
audio	Should audio files be returned.	-
video	Should video files be returned.	-
image	Should images be returned.	-
message	Should messages be returned.	-
document	Should folders be returned.	-
folder	Should folders be returned.	-
name=value	Should items with filename EQUAL to value be returned	Any string.

Each of these criteria can be inverted by prefixing it with "!".

This is equivalent to placing logical NOT before criterion.

Example:

- get not deleted, not marked as system images or videos:

```
filterAnd=AND:!deleted,!sys;OR:image,video
```

  - Scheme:

```
(!deleted AND !sys) AND (image OR video)
```

- get deleted videos or not deleted folders:

```
filterOr=AND:deleted,video;AND:!deleted,folder
```

  - Scheme:

```
(deleted AND video) OR (!deleted AND folder)
```

Examples

- Get 30 items from device xmofgp-1woobu-6dfc8n in path /TeamDrives from snapshots until 2018-06-19T11:14:02.658Z which are folders or audio or video files or messages and are sorted in a way that folders go first and then results are sorted by date:

```
/bsearch?count=30&device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FTeamDrives&snaptime=2018-06-19T11:14:02.658Z&sortTerms=folder,date&startIndex=0&filterOr=OR:folder,audio,video,message
```

- Get 30 items from device xmofgp-1woobu-6dfc8n in path /TeamDrives and its subfolders from snapshots until 2018-06-19T11:14:02.658Z which are deleted folders:

```
/bsearch?count=30&device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FTeamDrives&snaptime=2018-06-19T11:14:02.658Z&startIndex=0&filterAnd=AND:deleted,folder&recursive=1
```

- Get 30 items from device xmofgp-1woobu-6dfc8n in path /Users/ from snapshots until 2018-06-19T11:14:02.658Z which are videos. ALL videos will be returned, including deleted ones:

```
/bsearch?count=30&device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FUsers%2F&snaptime=2018-06-19T11:14:02.658Z&startIndex=0&filterAnd=AND:video
```

- Get 100 items from device xmofgp-1woobu-6dfc8n in path /TeamDrives that are in snapshots from 2018-06-09T21:00:00.999 till 2018-06-10T20:59:59.999 including previous versions of every item and do not include items' body:

```
/bsearch?device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FTeamDrives&includeBody=0&fullHistory=1&snaptimeFrom=2018-06-09T21:00:00.999Z&snaptimeTo=2018-06-10T20:59:59.999
```

- Get 100 items from device xmofgp-1woobu-6dfc8n in path /Users/ from snapshots until 2018-06-19T11:14:02.658Z which contain 'test' or similar words in filename or meta data and are NOT deleted and NOT marked as system:

```
/bsearch?device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FUsers%2Fs&snaptime=2018-06-19T11:14:02.658Z&searchTerms=test&filterAnd=AND:!deleted,!sys
```

- Get 100 items from device xmofgp-1woobu-6dfc8n in path /Sites/ from snapshots until 2018-06-19T11:14:02.658Z that are folders and are NOT deleted and NOT marked as system. Get these 100 items starting from 30th result entry and sorted by title:

```
/bsearch?device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FSites%2Fs&snaptime=2018-06-19T11:14:02.658Z&filterAnd=AND:!deleted,!sys,folder&startIndex=30&sortTerms=title
```

- Get 100 items from device xmofgp-1woobu-6dfc8n in path /Sites/ from snapshots until 2018-06-19T11:14:02.658Z that are images or videos or audios or messages or folders and are NOT deleted and NOT marked as system. Get these 100 items sorted by title:

```
/bsearch?device=xm0fgp-1w0obu-6dfc8n&pathRoot=%2FSites%2Fs&snaptime=2018-06-19T11:14:02.658Z&filterAnd=AND:!deleted,!sys;OR:image,video,audio,message,folder&sortTerms=title
```

Cloud Connector(s) Scheduling

About cloud connectors scheduling

This is a subsystem (aka SCCEF) responsible for reliable and manageable execution of backup and restore jobs of cloud connectors.

More information about jobs can be found here Job scheduling API.

This section describes all endpoints exposed by SCCEF API.

Currently, Cloud Connector Launcher (CCL) supports three signals that force finishing of work:

1. SIGTERM - will force quit CCL without cancelling any jobs.
2. SIGUSR1 - will force quit CCL and cancel all backup jobs.
3. SIGUSR2 - will force quit CCL and cancel all jobs.

Method GET /job_schedule/next_job

Returns the next available for execution job or 404 if there're no pending jobs.

This is the most important endpoint used primarily by CCL. Launchers are just simple dispatchers of jobs obtained from this endpoint to cloud connectors.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
}
```

The next endpoints are used primarily for debugging/QA purposes.

Method GET /job_schedule/jobs/ready

Returns the list of jobs ready to be dispatched for execution, e.g. jobs ready to be returned by GET /job_schedule/next_job endpoint.

By default jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Method GET /job_schedule/jobs/cancelled

Returns the list of cancelled (but not failed yet) jobs.

By default, jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Method GET /job_schedule/jobs/dispatched

Returns the list of dispatched (but not started/cancelled/failed yet) for execution jobs. e.g. jobs sent to CCL, but not executed yet.

By default jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Method GET /job_schedule/jobs/inprogress
Returns the list of jobs in progress. Job is considered in progress when it's started but not cancelled/succeeded/failed.

By default, jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Method GET /job_schedule/jobs/failed
Returns the list of failed jobs.

By default, jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Method GET /job_schedule/jobs/succeeded

Returns the list of succeeded jobs.

By default, jobs are returned for a period of time [now - 24 hours, now]. Headers 'from-time' and 'to-time' are used to override default period.

The response document will adhere to:

```
element job {
 element guid { guid }
 & element user-guid { guid }
 & element device-guid { guid }
 & element description { text }
 & element execsummary { text }?
 & element type { text }?
 & element priority { integer }
 & element active { boolean }
 & element start { iso8601-timestamp }
 & element scheduled { iso8601-timestamp }
 & element lifetime { iso8601-period }?
 & element dispatched { iso8601-timestamp }?
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & element failed { iso8601-timestamp }?
 & element succeeded { iso8601-timestamp }?
 & element progress { real }? # 0-1
}
```

Devices access API

Method GET /users/{user-id}/devices/{device-guid}/status

Returns the most recent status log entries for the device.

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Reply

The reply document adheres to the following schema:

```
element devstatus {
 element entry {
  element tstamp { timestamp }
  & element status { integer }
  & element adata {
      ecode { text } ? # error code if present
      progress { text } ? # generic upload progress (i.e. in %)
      objectsuploaded { integer } ? # support for progress. Connectors may set
"progress"
      objectstotal { integer } ?    # or objectsuploaded/objectstotal
    }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	OK	FAIL

Method POST /users/{user-id}/devices/{device-guid}/status

Post new status update for this device.

Please note, this rudimentary device status interface should only be used to log SIGNIFICANT events relating to backup functionality. We should only log the following types of events:

- A backup is scheduled (including the zulu time at which it is scheduled)
- A backup is initiated
- A backup is completed or aborted
- etc

Request

The request document must adhere to this schema:

```
element devstatus {
  element status { integer }
  & element adata {
      ecode { text } ? # error code if present
      progress { text } ? # generic upload progress (i.e. in %)
      objectsuploaded { integer } ? # support for progress. Connectors may set
"progress"
      objectstotal { integer } ?    # or objectsuploaded/objectstotal
    }
}
```

The back end will automatically time stamp the entry and clean up in entry history to only retain the last 1 or 2 entries.

Usually we store one last entry, but we need the second to preserve last useful action before connector went idle.

Used statuses

Status	Meaning
0	**Idle. Connector does nothing useful. Mainly applicable to desktop and server connectors.**
1	**Uploading**
2	**Downloading**
3	**Snapshot created**
4	**Scanning**
5	**Restoring**
6	**Restore complete**

Desktop and server connectors must send active status every 15 minutes. This doesn't apply to cloud connectors (since they are scheduled by backup manager and usually go offline after finishing their job) and mobile clients (since waking up the device every 15 minutes will quickly drain the battery).

Error codes are string values for better understanding.

Used error codes

Error code	Meaning
E_START	Failed to start backup
E_CONFIG	Connector configuration is invalid
E_TERM	Connector was terminated
E_RESOLVE	Failed to resolve remote host address
E_CONNECT	Failed to connect to remote host
E_LOGIN	Failed to login (username or password are incorrect)
E_READ	Failed to read data from the remote host

Error code	Meaning
E_WRITE	Failed to write data to the remote host
E_READ_API	Failed to read data from the API server
E_WRITE_API	Failed to write data to the API server
E_REVOKED	Previously granted permission has been revoked
E_EMPTY_ROOT	The root container is empty
E_OVERSIZE_ROOT	The root container is larger than one chunk

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	FAIL	OK	OK	FAIL

Method GET /users/{user-id}/devices/{device-guid}/health
Returns the most recent health information of the device.

Reply
The reply document adheres to the following schema:

```
element devhealth {
 element health { text } # health string
}
```

Possible health values

- healthy
- unhealthy
- critical

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET|PUT /users/{user-id}/devices/{device-guid}/history/range
Returns a range of snapshots on a given device. The method is really a GET, but to work around a deficiency in popular browsers, the PUT method can be used as well to the same effect.

Whenever you need to know parts of the snapshot history of a device, this is the endpoint to use. You can use /first and /latest to retrieve the possible extremes, and then use /range to get subsections of that range.

Performance-wise, the request will be faster if the resolution parameter is omitted, but this is of course not suitable for all applications.

As the rest of /history/*/ endpoints this one counts in accounts backup retention settings of a user as well as it keeps promise of 'at least 10 snapshots' even if some fall off the retention period.

Request

This request body must adhere to the following schema

```
element range {
  element start { iso8601-timestamp }  # range start
& element span { iso8601-timespan } # range span
& element count { integer } # max. no of snapshots to return - integer must be less
than 100
& element resolution { iso8601-timespan }?  # minimum interval between returned
snapshots (incurs performance overhead)
& element type { 'p' or 'c' }?  # optionally return only partial or complete snapshots
& element reverse?  # if present, reverse the query - cannot be used if resolution is
used
}
```

Note that the empty reverse element will reverse the direction of the query, if present. In this case, "start" is the newest date to consider, and "span" will span into the past (into older items). A query where "resolution" is used cannot be reversed.

Reply

The reply is a document with backup history entries for the device including timestamp and object root id.

Reply body schema

```
element history {
 element backup {
  element tstamp { timestamp }
  & element root { text } ?
  & element type { text }    # Type of snapshot: "p" - partial, "c" - complete
  & element size { integer } # Size of referenced data set in bytes
  & element account { text } ? # guid of the account created snapshot
 }
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Snapshot is not returned for administrator and partner accounts unless access request is approved.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET|PUT /users/{user-id}/devices/{device-guid}/history/count

Similar to "range" request but returns number of snapshots instead.

Request

This request body must adhere to the following schema

```
element range {
  element start { iso8601-timestamp }  # range start
& element span { iso8601-timespan } # range span
& element resolution { iso8601-timespan }?  # minimum interval between returned
snapshots (incurs performance overhead)
& element type { 'p' or 'c' }?  # optionally return only partial or complete snapshots
}
```

Reply

The reply is a document with backup history entries for the device including timestamp and object root id.

Reply body schema

```
element history {
 element count { integer }
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/devices/{device-guid}/history/first

Returns the info about the earliest 'complete' snapshot of a given device

As the rest of /history/*/ endpoints this one counts in accounts backup retention settings of a user as well as it keeps promise of 'at least 10 snapshots' even if some fall off the retention period. So the first snapshot may be dated older than the beginning of the retention period.

Reply

The reply is a document with first backup history entry for the device including timestamp and object root id.

Reply body schema

```
element history {
 element backup {
  element tstamp { timestamp }
  & element root { text } ?
  & element type { text }    # Type of snapshot: "p" - partial, "c" - complete
  & element size { integer } # Size of referenced data set in bytes
  & element account { text } ? # guid of the account created snapshot
 }?
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Snapshot is not returned for administrator and partner accounts unless access request is approved.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/devices/{device-guid}/history/latest

Returns the info about the latest 'complete' snapshot of a given device

As the rest of /history/*/ endpoints this one counts in accounts backup retention settings of a user as well as it keeps promise of 'at least 10 snapshots' even if some fall off the retention period. So the first snapshot may be dated older than the beginning of the retention period.

Reply

The reply is a document with latest backup history entry for the device including timestamp and object root id.

Reply body schema

```
element history {
 element backup {
  element tstamp { timestamp }
  & element root { text } ?
  & element type { text }    # Type of snapshot: "p" - partial, "c" - complete
  & element size { integer } # Size of referenced data set in bytes
  & element account { text } ? # guid of the account created snapshot
 }?
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Snapshot is not returned for administrator and partner accounts unless access request is approved.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/devices/{device-guid}/history/latest_imported

Returns the info about the latest snapshot of a given device imported by BSearch. If "x-completed-only" header contains "true" then returns info of the latest completed snapshot imported by BSearch or 404 error code if not found.

Reply

The reply is a document with latest backup history entry for the device including timestamp and object root id.

Reply body schema

```
element backup {
 element root { text }
 element tstamp { timestamp }
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Snapshot is not returned for administrator and partner accounts unless access request is approved.

Method GET /users/{user-id}/devices/{device-guid}/history/{snapshot-id}

Returns the info about the specified snapshot of a given device.

Reply

The reply is a document with first backup history entry for the device including timestamp and object root id.

Reply body schema

```
element history {
 element backup {
  element tstamp { timestamp }
  & element root { text } ?
  & element type { text }    # Type of snapshot: "p" - partial, "c" - complete
  & element size { integer } # Size of referenced data set in bytes
  & element account { text } ? # guid of the account created snapshot
 }?
}
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Snapshot is not returned for administrator and partner accounts unless access request is approved.

Method GET /users/{user-id}/devices/{device-id}/history/{object-hash}/root/{path}

Similar to the browsing of shares, this endpoint will return either an XML document with the directory contents at the path specified. It will not download the file specified, if the path points to an object - for this you must use the /download/ endpoint.

However, in order to generate a URI for the download, you must use the "alias" element provided in this response. The "alias" is an alias for an object hash which can be used for a limited period of time in the /download/ request.

So, if you are passed an alias of "asdf123" in the response document and the directory contains a file "here.text", you can download this file from /download/asdf123/here.text. This download will not require authentication. The alias, however, has a limited lifetime and will expire 30 minutes after the last time the directory contents were retrieved.

The "treesize" is the number of bytes that is referenced by this directory directly or indirectly (meaning, the number of bytes referenced by the entire tree structure rooted in this directory).

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

Reply

The reply document in case the path points to a directory is:

```
element directory {
 element alias { text }
 & element name { text }
 & element owner { text }
 & element treesize { integer }
 & (element file {
     element name { text }
     & element modified { text }?  # Not all connectors set modified time
     & element size { integer }
    }
    | element directory {
      element name { text }
      & element treesize { integer }?   # We may or may not have a size on sub
directories
      }
    )*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /devices

Returns system-wide list of devices satisfying supplied criteria.

Request document schema

```
element filter {
 element kind { text } # pc, cloud, drive, system
 & element type { text }?   # windows, osx, android, ios, o365ng, etc
 & element limit { integer } # number of records to return
 & element offset { integer } # for pagination
 & element all { boolean }? # include deleted devices, include devices for deleted
accounts
}
```

On success a 200 OK is reported.

Response document schema

```
element devices {
  element device {
    element guid { guid }
    & element account { guid }  # guid of device owner
    & element kind { text } # cloud, pc, drive, system
    & element type { text } # windows, o365ng, etc
    & element name { text }
    & element created { timestamp }?
    & element deletion-deadline { timestamp }?
    & element backup-retention { timespan }?
    & element backup-retention-updated { timestamp }? last update of the device
specific snapshot retention
 }*
}
```

U/EnumerateDevices *ACL* is required.

Method GET /users/{user-id}/devices?all={0|1}

Returns a list of registered devices and devices scheduled for deletion for the given account if all set to 1.

Reply

The reply should be a 200 OK and a body with the following schema:

Response document

```
element devices {
 element pc {
  element guid { text }
  & element created { timestamp }?
  & element deletion-deadline { timestamp }?
  & element backup-retention { timespan }? # device-specific backup retention period
  & element backup-retention-updated { timestamp }? last update of the device specific
snapshot retention
  & element name { text }
  & element type { text }?  #windows, osx, android, ios
  & element fromgroup { guid }*
  & element centralconfig { guid } ?
 }*
 & element cloud {
    element guid { text }
    & element created { timestamp }?
    & element deletion-deadline { timestamp }?
    & element backup-retention { timespan }? # device-specific backup retention period
    & element backup-retention-updated { timestamp }? last update of the device
specific snapshot retention
    & element name { text }
    & element type { text }
    & element uri { text }
    & element login { text }? # login is returned only for system users
    & element password { text }? # password is returned only for system users
    & element fromgroup { guid }*
 }*
 & element drive {
    element guid { text }
    & element created { timestamp }?
    & element deletion-deadline { timestamp }?
    & element backup-retention { timespan }? # device-specific backup retention period
    & element backup-retention-updated { timestamp }? last update of the device
specific snapshot retention
    & element name { text }
    & element fromgroup { guid }* # list of groups where this our shared drive is
present
 }*
 & element remotedrive {  # this is a 'drive' shared to the current account by another
account
    element guid { text }
    & element created { timestamp }?
    & element deletion-deadline { timestamp }?
    & element backup-retention { timespan }? # device-specific backup retention period
    & element backup-retention-updated { timestamp }? last update of the device
specific snapshot retention
    & element name { text }
    & element fromgroup { guid }* # list of groups this remotely shared drive belongs
to
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method POST /users/{user-id}/devices/

Details (such as user readable name) are provided in the body of the request. This request is used to create a new device from which backups can be made. For now, these devices are "sources" of data - but in the future they will likely be destinations as well once synchronization is introduced.

For now we have two types of devices; "pc" devices and "cloud" devices. The PC device is the normal PC client for which we only supply a descriptive name. The "cloud" device is, for example, an Office 365 account for which we also need to supply a cloud service type, a URI, a login and a password.

Request body schema

```
element pc {
 element name { text }
 & element backup-retention { timespan }? # device-specific backup retention period
 & element type { text }?   #windows, osx, android, ios
 & element attributes {
  element attribute {
   element name { text }
   element value { text }
  }*
 }?
}
| element cloud {
 element name { text }
 & element type { text }    #o365, ftp, gdrive, skydrive
 & element backup-retention { timespan }? # device-specific backup retention period
 & element uri { text }?
 & element login { text }?
 & element password { text }?
 & element attributes {
  element attribute {
   element name { text }
   element value { text }
  }*
 }?
}
| element drive {
  element name { text }
  & element backup-retention { timespan }? # device-specific backup retention period
  & element attributes {
   element attribute {
    element name { text }
    element value { text }
  }*
 }?
}
```

Reply

On success, code 201 (Created) is returned and the Location header will hold the URI of the created entry.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/devices/{device-guid}

Returns information about the device.

Reply

The reply should be a 200 OK and a body with the following schema:

Response document

```
element device {
  element kind { text } # pc, cloud, drive, system, etc
  & element name { text }
  & element type { text } # device subtype
  & element created { timestamp }
  & element deletion-deadline { timestamp }?
  & element backup-retention { timespan }? # device-specific backup retention period
  & element backup-retention-updated { timestamp }? last update of the device specific
snapshot retention
  & element centralconfig { guid }? # if device has central config assigned
  & element fromgroup { guid }? # group id this devices comes from
  & element uri { text }? # some cloud devices have it
  & element login { text }? # some cloud devices have it
  & element password { text }? # some cloud devices have it
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method PUT /users/{user-id}/devices/{device-guid}

This method is used to update a named device.

The input document allows both for name change and even type change.

Request body schema

```
element pc {
 element name { text }?
 & element backup-retention { timespan }? # device-specific backup retention period
 & element type { text }?   #windows, osx, android, ios
}
| element cloud {
   element name { text }?
   & element backup-retention { timespan }? # device-specific backup retention period
   & element type { text }?
   & element uri { text }?
   & element login { text }?
   & element password { text }?
}
| element drive {
   element name { text }
   & element backup-retention { timespan }? # device-specific backup retention period
}
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method DELETE /users/{user-id}/devices/{device-guid}

This method deletes the named device. If delete-retention-period resource is set than device will be scheduled for deletion instead of deleting directly.

On success, a 200 OK response is given.

If the device-guid given is a 'remotedrive' type device, the actual device is not deleted, but the current account guid is removed from the list of users that the given drive device is shared to.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /users/{user-id}/devices/{device-guid}/revive

This method is used to resurrect scheduled for deletion device.

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method POST /users/{user-id}/devices/{drive-guid}/peers

This will add a sharing entry to the given drive. Drives can be shared with other users, as it is today. But in the future we may want to allow groups of users, or we may want read-only sharing or time limited sharing. This API will be able to serve those needs.

Request body schema

```
element account {
 element guid { account-guid }
 | element aname { account-token-aname } # aname of 'p','r','u' type token
 }
```

Reply

On success, code 201 (Created) is returned and the Location header will hold the URI of the created entry.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/devices/{drive-guid}/peers

This will display the current sharing configuration for a sync device (and possibly, in the future, for other device types also - who knows).

We only support account sharing now; but in the future we may support groups or other sharing constructs.

Request body schema

```
element peers {
 element account {
    element guid { account-guid }   # peer account GUID
  & element fullname { text }?    # primary contact fullname of peer account, if any
  & element email { text }?       # primary contact e-mail of peer account, if any
  & element owner { boolean }?    # is account drive owner
 }*
}
```

Reply

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method DELETE /users/{user-id}/devices/{drive-guid}/peers

Calling user will leave the share.

Reply

On success, code 200 (OK) is returned.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method DELETE /users/{user-id}/devices/{drive-guid}/peers/account/{peer-account-guid}

This will delete the peer account entry given.

Reply

On success, code 200 (OK) is returned.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method POST /users/{user-id}/devices/{device}/history/

This method is used by the clients whenever they have completed a backup.

Request

The body of the request will define the hash of the backup set root. The timestamp is the UTC time on the client device at the time the system was snapshot, if snapshots are in use (on Windows for example). If snapshots are not used (on MacOS for example), the timestamp should match the initiation of the backup.

```
element backup {
 element root { hash }
 & element lastroot { hash }?  # See comment!
 & element timestamp { timestamp }
 & element type { text }?    #Optional. Type of snapshot: "p" - partial, "c" - complete
(default)
}
```

The "lastroot" item is required for SYNC type devices, and must be set to the previous root hash - only if there is no history on the device may a SYNC type device POST an item without a lastroot item. This ensures that updates cannot race between connectors. The "lastroot" item should not be used on any other device type.

If "lastroot" is supplied and the given hash is not equal to the most recent hash in the history, the request will fail with a 409 Conflict error.

Reply

The reply should be a 201 (Created) - the URI of the created backup object will be given in the Location header.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	FAIL	OK	OK	FAIL

Method POST /users/{user-id}/devices/{device}/auth_code/

This method is used to exchange an OAuth2 authorization code to access, refresh token and store them on Veritas SaaS Backup servesr to use by cloud backup agents.

Request

```
element auth_code {
 element type { text } # "google"
 element code { text } # authorisation code received in web callback
 element redirect_uri { text } # One of Authorized Redirect URIs as was registered in
Google App settings
}
```

Reply

The reply should be a 201 (Created).

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /users/{user-id}/devices/{device}/attributes

This method is used list all attributes assigned to a given user device.

Response

```
element attributes {
 element attribute {
  element name { text }
  & element value { text }
 }*
}
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method GET /users/{user-id}/devices/{device}/attributes/{key}

This method is used to get value for specified key of named device.

The value is UTF8 string.

Example

```
GET /users/asdf-2345-1233-asdf/devices/fdsa-fdsa-fdsa-
fdas/attributes/google_refresh_token

200 OK
content-type: application/application/octet-stream
content-size: 31

mwV1vtp6wKfhAfkMRFp-1BRdCwl9.cvGF
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method PUT /users/{user-id}/devices/{device}/attributes/{key}

This method is used to create/update value for specified key of named device.

The value is UTF8 string.

Example

```
PUT /users/asdf-2345-1233-asdf/devices/fdsa-fdsa-fdsa-
fdas/attributes/google_refresh_token
content-type: application/octet-stream
content-size: 31

mwV1vtp6wKfhAfkMRFp-1BRdCwl9.cvGF
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method DELETE /users/{user-id}/devices/{device}/attributes/{key}
This method is used to delete specified key of named device.

Example

```
DELETE /users/asdf-2345-1233-asdf/devices/fdsa-fdsa-fdsa-
fdas/attributes/google_refresh_token
```

On success a 200 OK is reported.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	OK	FAIL

Method POST /users/{user-id}/devices/{device-guid}/current/{path}
This endpoint is used for form-based file uploads. Any drive-type device can be used in the above. The following example shows how CURL can be used to upload a file:

```
curl -F "file=@wc2.pl" \
  'http://joe:*****@eagle:8080/users/add227e7-7f01-4641-803a-
864a920aa816/devices/0utjdm-tow0qw-lu51tw/current/'
```

The FILE type form field must be named "file". The form must result in a multipart/mixed type document to be POST'ed.

The file will be merged into the most recent snapshot on the device. The POST may return Conflict both if the mount was changed while the upload was happening, or if the file already exists in the directory specified.

For now, the implementation of this endpoint is not ready for general use. It will not properly handle large file uploads. File up to 10-20 megabytes should be fine.

This endpoint is also used for directory creation in online drives. Form field must be named "directory" following directory name. Example directory creation request:

```
curl --user 'ipuser@mail.com:123' -F "directory=TESTDIR"
'https://saasbackup.veritas.com/users/j87wjp-7xxphl-45jc8b/devices/u4v0kd-z97g1u-
ro3ezw/current'
```

If device is shared with current user but group policy doesn't allow to access it 403 error code is returned. Client code must handle it properly and display 'Insufficient permissions' message.

This endpoint is also accepts several additional parameters:

- file.directory - specifies directory name to be created in online drive
- file.setname - specifies alternative name for the file being uploaded
- file.mtime - specifies file modification time
- file.overwrite - overwrite file silently instead of failing with 409 in case if file already exists

Method PUT /users/{user-id}/devices/{device-guid}/current/{path}
Moves file from one location in online drive to another. Simply renames file if source and target directory are the same.

Request

```
element move {
 element device { guid } # guid of target device
 & element path { text } # full path in the target device including target filename
starting from /
 & element overwrite { boolean }? # overwrite existing file instead of returning 409
}
```

Returns 200 OK on success.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method DELETE /users/{user-id}/devices/{device-guid}/current/{path}

This endpoint allows to delete files and folders from online drive.

Example

```
DELETE /users/j87wjp-7xxphl-45jc8b/devices/70sd4l-lmlb7e-zcu53e/current/444.jpeg
```

Returns 200 OK on success.

Central configuration subdocument

A number of API endpoints will work with the central configuration subdocument which is defined as follows:

```
configsubdoc =
 element config {
  element user_selection {bool}
  & element includes {
     element location {
      element path { text }
      & element enabled { bool } ?
     }*
    }
  & element excludes {
     element location {
      element path { text }
      & element enabled { bool } ?
     }*
    }
  & element autoRules {
      element name { text }
      & element rule { text }
      & element id { text }
      & element types { text }
      & element value { text }
    }*
  & element backup_window {
     element hard ?                                     # only present on hard
windows; window is soft by default
     & element start { iso8601-timestamp-without-tz }     # Timestamp without timezone
designator  "20:44:00" for example
     & element extent { iso8601-duration }              # "PT4H" for example
  }?
}
```

Method GET /users/{user-id}/centralconfig

Returns the list of central configurations created by the user.

Reply

The reply document adheres to the following schema:

```
element configurations {
 element configuration {
  element guid { text }
  & element name { text }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user-id}/centralconfig

Creates a new empty central configuration.

Request

The request document must adhere to the following schema:

```
element configuration {
  element name { text }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/centralconfig/{centralconfig-id}

Returns central configuration including name and configuration itself. Config subdocument isn't used by the server and can contain any valid XML structure. Clients use scheme described below.

Reply

The reply document adheres to the following schema:

```
element configuration {
  element name { text }
  & configsubdoc
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method PUT /users/{user-id}/centralconfig/{centralconfig-id}

Updates central configuration.

Request

The request document must adhere to the following schema:

```
element configuration {
  element name { text }
  & configsubdoc
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user-id}/centralconfig/{centralconfig-id}

Deletes central configuration.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/devices/{device-id}/centralconfig

Reply

The reply document adheres to the following schema:

```
element configuration {
  element name { text }
  & configsubdoc
}
```

Method POST /users/{user-id}/devices/{device-id}/centralconfig

Binds device to given central configuration. Device can have only one configuration assigned.

Request

The request document must adhere to the following schema:

```
element configuration {
  element guid { text }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user-id}/devices/{device-id}/centralconfig

Unbinds device from central configuration.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/centralconfig/{centralconfig-id}/devices

Returns information about all devices using current central configuration.

Reply

The reply document adheres to the following schema:

```
element devices {
  element device {
    element guid { text }
    & element owner {
      element guid { text }
      & element name { text }
      & element email { text } ?
    }
  }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user-id}/centralconfig/{centralconfig-id}/rules

Add device auto-add rule to central configuration. I.e. new device will be automatically added centralconfig if it is of certain type(win, osx, etc) and user is a member of a certain group (or user explicitly specified)

Request

The request document must adhere to the following schema:

```
element rule {
  element name { text }
  & element group { text } # GUID of a group user is member of
    | element account { text } # or explicit account GUID
  & element devtype { text }* #windows, osx, android, ios
}
```

On success a 201 OK is returned.

"Location" header contains path to the new entry created.

Example

```
curl -u gadmin@test.com:qwe -k -v -X POST 'https://localhost:8501/users/1ta3k6-8m7fdo-
brss5w/centralconfig/vbbi2e-c661jf-zgss0v/rules' -d '<?xml version="1.0" encoding="utf-
8"?><rule><name>rule1</name><group>28ze55-d1mbxv-
wypoqn</group><devtype>windows</devtype></rule>'

HTTP/1.1 201 Created
location: /users/1ta3k6-8m7fdo-brss5w/centralconfig/vbbi2e-c661jf-zgss0v/rules/9mvlds-
p5525f-h62ym7
content-length: 0
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/centralconfig/{centralconfig-id}/rules

Get list of device auto-add rules for specified central configuration.

Reply

The reply document adheres to the following schema:

```
element rules {
  element rule {
    element name { text }
    & element group { text }? # GUID of a group user is member of
    & element account { text }? # or explicit account GUID
    & element devtype { text }* #windows, osx, android, ios
  }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method DELETE /users/{user-id}/centralconfig/{centralconfig-id}/rules/{rule-id}

This endpoint allows to delete specified auto-add rule.

Example

```
curl -u gadmin@test.com:qwe -k -v -X DELETE 'https://localhost:8501/users/1ta3k6-
8m7fdo-brss5w/centralconfig/vbbi2e-c661jf-zgss0v/rules/9mvlds-p5525f-h62ym7'
```

Returns 200 OK on success.

Method GET /users/{user-id}/devices/{device-id}/resources

Returns the resource consumption reported by a device.

Reply

The reply document adheres to the following schema:

```
element resources {
  element resource {
    element name { text }
    & element usage { text } # usage reported by the connector.For now it's just an
integer values (seats)
  }*
}
```

G/DevResources *ACL* is required in order to access this endpoint

Method PUT /users/{user-id}/devices/{device-id}/resources/{resource-name}/usage

Reports current usage of the resource "resource-name" by the device. This is necessary for so called client-side resources which don't have server-side representation. Using this endpoint, we can tell the server current consumption and eventually calculate overall account consumption.

Request

The request document must adhere to the following schema:

```
element usage { integer } # currently we support only integer resources
```

Returns 200 OK on success. Returns 404 if resource-name is unknown.

U/DevResourceUsage *ACL* must be enabled for the account to be able to access this endpoint.

Favourites API

Favourites are shared among all clients accessing the user data. Favourites can be used to "guide" the user to the most likely data he needs to access - for example, in the case of the desktop PC backup agent, it would make sense for the backup agent to automatically add the users home directory and/or desktop, documents, etc. to the favourites list, to save the user from the trouble of browsing there.

A favourite item simply consists of a device id, label and an object path - we can favourite both files and folders (and other objects as we add support for them).

This means, favourites work across snapshots.

The path to the favourite object is a sequence of directory entry names separated by the slash character.

NOTE: We always use the slash character for separating entries, even if the item we favourite is originating from a windows system that internally uses backslash as separating character.

Method POST /favourites

This method allows the authenticated user to add a favourite.

Request body

The request body must adhere to the following schema

```
element favourite {
 element device { text } # guid of the device
 & element snapshot { text } ? # id of the snapshot file is favourited in
 & element path { text }
}
```

Both the device and path must be non-empty.

There is no validation of the path, and if the path is removed from future backup sets, there may be "dangling" favourites that point to paths that no longer exist. The user agents must be able to deal with this.

If snapshot is absent file is favourited in the latest snapshot.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /favourites

This method allows the authenticated user to retrieve all favourites.

Response body

The response body will adhere to the following schema

```
element favourites {
 element favourite {
  element device { text }
  & element snapshot { text }? # id of the snapshot file is favourited in
  & element label { text }
  & element path { text }
  & element dname { text }? # well readable alias for the favourite
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method DELETE /favourites/{label}

This method allows for deletion of previously created favourites. The URI used in this request is the one returned in the Location header from the POST.

On success a 200 response is sent.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

FCM Recipients API

Method GET /users/{user-id}/fcm/recipients

Returns list of FCM (Firebase Cloud Messaging) recipients for given account

Response

```
element recipients {
 element recipient {
  element id { text } # recipient guid
  & element token { text } # push notification token
  & element identifier { text }? # device identifier
  & element name { text }? # device name
  & modified { iso8601-timestamp } # last modified date
 }*
}
```

ACL

G/FCMRecipients

Method GET /users/{user-id}/fcm/recipients/{recipient-id}

Returns FCM recipient details

Response

```
element recipient {
 element id { text } # recipient guid
 & element token { text } # push notification token
 & element identifier { text }? # device identifier
 & element name { text }? # device name
 & modified { iso8601-timestamp } # last modified date
}
```

ACL

G/FCMRecipients

Method POST /users/{user-id}/fcm/recipients

Adds new FMC recipient.

Request

```
element recipient {
 element token { text } # push notification token
 & element identifier { text }? # device identifier
 & element name { text }? # device name
}
```

Response

The response will contain a location header with the URI of the newly created recipient

ACL

P/FCMRecipients

Method PUT /users/{user-id}/fcm/recipients/{recipient-id}

Updates details of given recipient

Request

Note: If an optional field is not provided then his value will be cleaned up.

```
element recipient {
 element token { text } # push notification token
 & element identifier { text }? # device identifier
 & element name { text }? # device name
}
```

ACL

U/FCMRecipients

Method DELETE /users/{user-id}/fcm/recipients/{recipient-id}

Deletes given recipient

ACL

D/FCMRecipients

Global Environment Attributes

The concept of GEAS (Global Environment Attributes Storage) is to be able to have attributes not only on the accounts and devices but also on the global level for the whole environment (dev/test/prod).

The global attributes form a tree and can be addressable by a path in the tree accessing /attributes endpoint.

The name of the attribute should only contain lower and upper case English letters, numbers and underscores. (a-z, A-Z, 0-9, _)

The value of the attribute is just a string.

conf	
bsearch	

nodes	44
threads	42
cc	
threads	20

GET

Use GET method to request an attribute or attributes subtree.

Access /attributes endpoint to get the whole attributes tree. Access /attributes/<attribute path> endpoint to query a specific attribute/subtree.

Reply codes: 200 if the specified attribute exists, 404 otherwise.

Method GET /attributes/conf/bsearch/threads

Request to get a specific attribute

Reply body

```
<threads>42</threads>
```

Method GET /attributes/conf/bsearch

Request to get attributes subtree

Reply body

```
<bsearch>
  <nodes>44</nodes>
  <threads>42</threads>
</bsearch>
```

Method GET /attributes

Request to get all attributes

Reply body

```
<attributes>
  <conf>
    <bsearch>
      <nodes>44</nodes>
      <threads>42</threads>
    </bsearch>
    <cc>
      <threads>20</threads>
    </cc>
  </conf>
</attributes>
```

PUT

Use PUT method to create/update an attribute or attributes tree. It deletes the specified subtree if any, and creates a new one described in the request body.

Reply codes: 201 on success.

Method PUT /attributes/conf/bsearch/threads

Request to update a single attribute

Request body

```
<threads>43</threads>
```

Result

Sets conf/bsearch/threads attribute to the specified value 43

conf	
bsearch	
nodes	44
threads	43
cc	
threads	20

Method PUT /attributes/conf/bsearch

Request to update attributes subtree

Request body

```
<bsearch>
  <threads>30</threads>
  <debug>true</debug>
</bsearch>
```

Result

Deletes existing conf/bsearch attributes sutree (i.e. deletes conf/bsearch/threads and conf/bsearch/nodes attributes), creates a new subtree described in the request body

conf	
bsearch	
threads	30
debug	true
cc	

threads	20

Method PUT /attributes

Request to update the whole attributes tree

Request body

```
<attributes>
  <conf>
    <bsearch>
      <nodes>44</nodes>
      <threads>42</threads>
    </bsearch>
    <cc>
      <threads>20</threads>
    </cc>
  </conf>
</attributes>
```

Result

Deletes all existing attributes, creates a new attributes subtree

conf	
bsearch	
nodes	44
threads	42
cc	
threads	20

DELETE

Use DELETE mathod to delete an attribute or attributes tree.

Reply code: 200 on success (does not check whether the specified attribute existed)

Method DELETE /attributes/conf/bsearch/nodes

Request to delete a single attribute

Result

conf	
bsearch	
threads	42

cc	
threads	20

Method DELETE /attributes/conf/bsearch

Request to delete attributes subtree

Result

conf	
cc	
threads	20

Method DELETE /attributes

Request to delete all attributes

Result

The whole attributes tree gets deleted.

Listen to updates of global attributes

Changes of global attributes tree are propagated by vqueues. Names of vqueues in use are attributes-<attribute-full-name> and attributes-*. Changes are propagated to the vqueue of the attribute specified in the endpoint, vqueues of all its parents, and a global attributes-* vqueue.

Method PUT, DELETE /attributes/conf/bsearch/threads

Update a single attribute

Triggered vqueues

- attributes-conf
- attributes-conf-bsearch
- attributes-conf-bsearch-threads
- attributes-*

Method PUT, DELETE /attributes/conf/bsearch

Update attributes subtree

Triggered vqueues

- attributes-conf
- attributes-conf-bsearch
- attributes-*

Method PUT, DELETE /attributes

Update all attributes

Triggered vqueues

- attributes-*

Job scheduling API

This page describes the Job Scheduling API and gives background information necessary to understand the purpose of job objects.

About job scheduling

A job object is a collection of one or more "commands" that must be executed on a given device.

When a device executes a job, each of the individual commands are executed sequentially, and each command executor is provided with the "deadline" for the job execution, the start+lifetime, if a lifetime for the job is provided. This is necessary for backup jobs to respect hard backup windows.

It would be good to be able to schedule jobs in advance of their execution. If only a few hours in advance. It would allow for the connectors to do "rare polling" instead of subscribing for events, or at least to have a long vMQ timeout.

The problem with scheduling in advance is we may have to retract an already published future job if the schedule changes. But this doesn't have to be a problem. Consider if the connector will:

1. Subscribe to the job-{device-id} volatile queue with a 1 hour timeout
2. On events or "reset", inquire about current job schedule
3. Track when the next job expires
4. Upon job expiry, register intent to execute job. If registration request fails, the job is no longer scheduled(!)
5. otherwise, job execution commences; the server will not unschedule a job that has been started. Upon job start, a progress record is sent to the server.
6. Any job that is executing must periodically (every few minutes) update its progress information. The server can fail this call to signal a job cancellation.

It is important that the connectors verify the accuracy of their local clocks periodically using the /time endpoint we provide. If the client clock is off by more than say 5-10 minutes, job scheduling will be inaccurate and will affect the ability of the system to properly schedule jobs within the windows given. A backup status report should warn about this.

Actual job contents

A job is a collection of commands. Typically, a job contains just a single command - and right now, our connectors will support only a single or a few commands. But this is a system that is ready for future expansion. The high-level idea is that of controlled remote command execution; the connector provides a safe and controlled set of commands that the backend can request to have executed at will.

One such command, would be "backup". Executing that command will run a backup.

The command sequence is provided as an XML subdocument; we use a simple SXML inspired format which basically allows us to use our existing XML parser as a language interpreter for our command set.

For example, the backup command which doesn't take any arguments can be executed as the only command in a job command sequence like this:

```
<commands>
 <backup/>
</commands>
```

We could also execute a job in our test suite that would execute two commands; one that counts to 10, then another that counts to 20

```
<commands>
 <count-to><limit>10</limit></count-to>
 <count-to><limit>20</limit></count-to>
</commands>
```

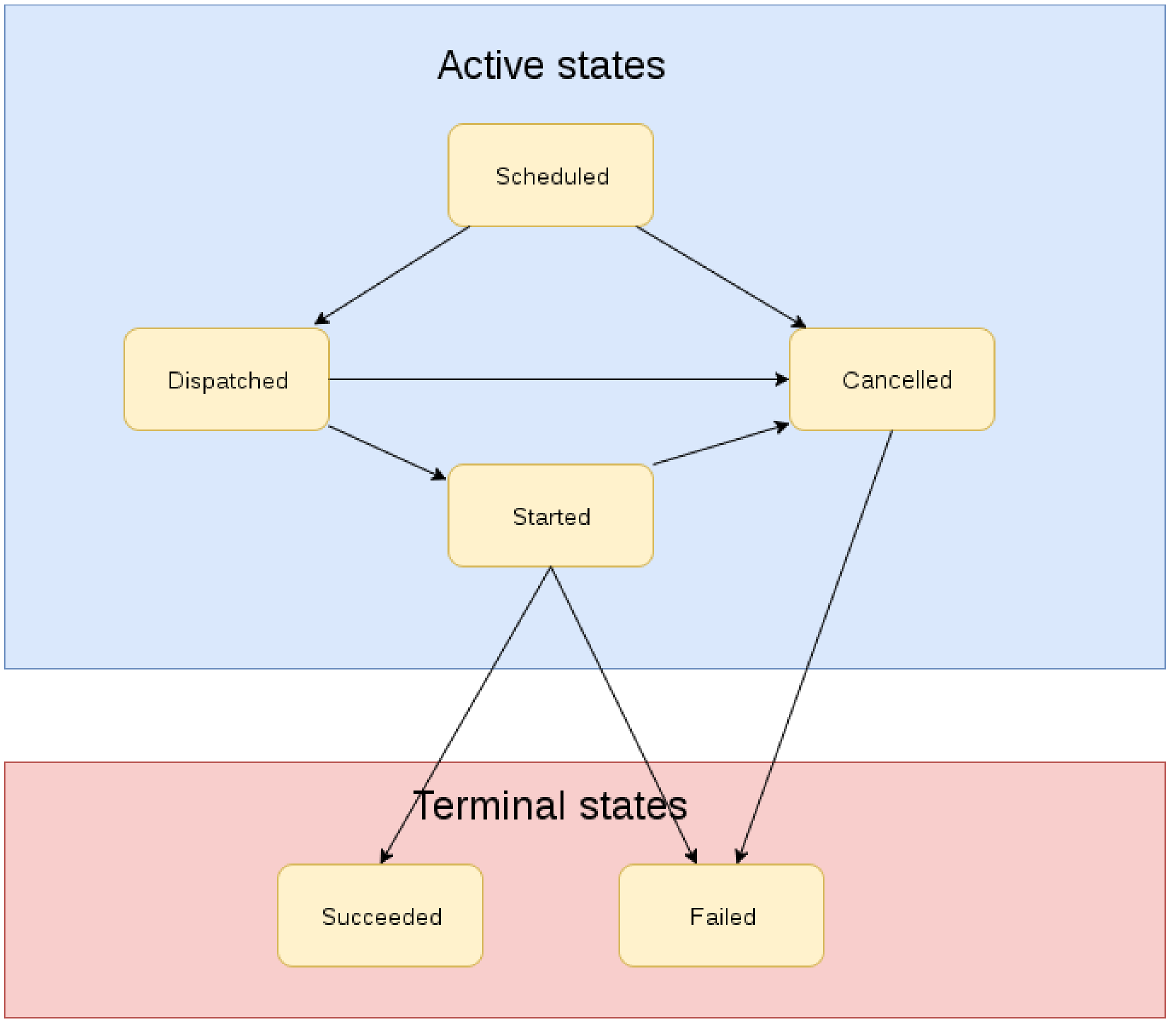
As can be seen, we use a simple scheme for named parameters that once again allows our existing XML parser and schema validator to function as a command interpreter with command parameter parsing and validation.

Every command in a job command sequence is executed in sequence. Job execution is completely sequential - we cannot start one job before the current one is complete. This allows us to tie both our own generated Trace events to the job object using the evlog framework, and, it allows us to (eventually) forward system log (operating system generated) events that occur while the job is executing, to the job object. This would, for example, allow us to tie operating system notifications about physical disk failure to a job object, thereby allowing a human operator much greater insight into why his backups may be failing.

Job lifecycle

For jobs to be correctly handled they need to follow some rules of their lifecycle. Rules may differ from system to system as long as they don't conflict with each other. In cloud connectors scheduling system aka SCCEF we use the next job lifecycle which works well and if preferred for other systems as well.

This lifecycle is explained better in SCCEF document.



Method GET /users/{user-id}/devices/{device-id}/jobs

This endpoint provides access to PAST, CURRENT and FUTURE jobs, both cancelled, failed and successfully completed ones. There will quickly be a very big number of jobs, so the endpoint will return all jobs with a "start" time set between now-P7D and now+P7D.

In order to query some other time window, the "job-window" header can be set to some other ISO8601-period.

To query only active jobs send header "active-jobs-only" with value "true".

Response

The response document will adhere to:

```
element jobs {
 element job {
  element guid { text }
  & element description { text }
  & element execsummary { text }? # some job execution statistics
  & element type { text }? # job type i.e. backup, restore, srestore
  & element priority { integer }
  & element active { boolean } # job is active until it's finished
  & element start { iso8601-timestamp }
  & element scheduled { iso8601-timestamp }
  & element lifetime { iso8601-period }?        # if executing in a hard window
  & element dispatched { iso8601-timestamp }?   # for cloud connector jobs - time when
job has been dispatched for execution
  & element started { iso8601-timestamp }?
  & element cancelled { iso8601-timestamp }?
  & ( element failed { iso8601-timestamp }
     | element succeeded { iso8601-timestamp })?
  & element progress { real }?   # only present if job is in progress
 }*
}
```

Method PUT /users/{user-id}/devices/{device-id}/jobs

Same as GET, returns jobs in a specified window, but parameters are passed in the request body.

Request

The request document must adhere to this schema:

```
element filter {
  element from-time { timestamp }
  & element to-time { timestamp }
  & element active-only { boolean }?
  & element extended-info { boolean }?
  & element stats {
     element name { text }*
    }?
}
```

Unlike GET this method allows to specify more flexible window.

Response

The response document will adhere to:

```
element jobs {
 element job {
  element guid { text }
  & element description { text }
  & element execsummary { text }? # some job execution statistics
  & element type { text }? # job type i.e. backup, restore, srestore
  & element priority { integer }
  & element active { boolean } # job is active until it's finished
  & element start { iso8601-timestamp }
  & element scheduled { iso8601-timestamp }
  & element lifetime { iso8601-period }?        # if executing in a hard window
  & element dispatched { iso8601-timestamp }?   # for cloud connector jobs - time when
job has been dispatched for execution
  & element started { iso8601-timestamp }?
  & element cancelled { iso8601-timestamp }?
  & ( element failed { iso8601-timestamp }
     | element succeeded { iso8601-timestamp })?
  & element progress { real }?   # only present if job is in progress
  & element ready { boolean }? # only present if extended-info was requested
  & element stats { # only present if stats were requested
     element stat {
      element name { text }
      & element value { text }
     }*
    }?
 }*
}
```

Method POST /users/{user-id}/devices/{device-id}/jobs

This request allows for creation of a new job. This is used by clients that do their own scheduling but still wish to be able to associate their operations with a job id (for MTrace job id tagging, job completion status etc.)

The job can be created either by supplying a "start" time (usually in the future), or by supplying "immediate" element, or by supplying the "started" element.

By supplying "immediate" it's guaranteed that job start time is set to scheduled time but job is not marked as started immediately but rather a little bit later by another system executing job.

In the last case, the job is created with a start and started time set to the current time.

```
element job {
 ( element start { iso8601-timestamp }      # either time when job execution should
start
   | element started                        # or job mark job as immediately started
"now" (used by some connectors)
   | element immediate)                     # equivalent to start { now() }, but uses
server time.
 & element description { text }            # short english description of job
 & element type { text }?                  # i.e. backup, restore, srestore
 & element priority { integer }?
 & subdocument commands?
}
```

This request will also issue a notification in job-* vqueue with type 'update' and the following data:

```
element job {
 element job-guid { guid }
 & element account-guid { guid }
 & element device-guid { guid }
 & element update { 'newjob' } # concrete value 'newjob'
}
```

To be able to create jobs for cloud connectors account must have the following ACLs enabled:

- P/CCScheduleSRestore - create single file/folder restore jobs
- P/CCScheduleRestore - create restore jobs

- P/CCScheduleBackup - create backup jobs as a user

Method GET /users/{user-id}/devices/{device-id}/jobs/{job-id}

This endpoint returns the full job details of any given job. This is the endpoint a connector will use prior to beginning execution of a job.

Response

The response document will adhere to:

```
element job {
 element guid { text }                          # job guid
 & element scheduled { iso8601-timestamp }      # time of job creation
 & element start { iso8601-timestamp }          # time when job execution should start
 & element type { text }? # job type. Can be any string. Valid cloud connector job
types: backup, restore, srestore
 & element lifetime { iso8601-period }?         # if executing in a hard window
 & element description { text }                 # short english description of job
 & element execsummary { text }?                # some job execution statistics
 & element dispatched { iso8601-timestamp }?    # for cloud connector jobs - time when
job has been dispatched for execution
 & element started { iso8601-timestamp }?
 & element cancelled { iso8601-timestamp }?
 & (element failed { iso8601-timestamp }
    | element succeeded { iso8601-timestamp })?
 & element progress { real }?                   # range 0-1
 & element priority { integer }                 # priority compared to other jobs
 & element active { boolean }                   # job is active until it's done
 & subdocument commands ?                       # commands document attached to current
job. May contain restore instructions for example
  }
```

Method PUT /users/{user-id}/devices/{device-id}/jobs/{job-id}

This endpoint must be used by the connector when it initiates execution of a job, and when it concludes a job execution either by failure or success.

Request

The request document must adhere to:

```
element job {
 (element started { iso8601-timestamp }
 | element cancelled { iso8601-timestamp }
 | element failed { iso8601-timestamp }
 | element succeeded { iso8601-timestamp }
 | element progress { real })  # range 0-1
 & element execsummary { text }? # some job execution statistics
}
```

When a connector initiates a job execution, it MUST put a "started" date on the job using this endpoint.

When the job is terminated, it MUST either put a "failed" or "succeeded" date on it.

When a job is cancelled, the back-end will already have set the "cancelled" time; it is up to the connector to actually cease execution ASAP.

Typically, we want to know *why* a job was cancelled or why it failed. This can be determined by searching the evlog for records tagged with the job id (guid).

The progress element can be PUT as many times as the connector likes; it would be advisable to PUT a new progress indication at regular intervals, say every couple of minutes perhaps. The progress records are stored along with time stamps so that we can present a reasonable progress indication on the web UI. The backend may prune progress indications for terminated jobs, to contain the amount of historical information kept in the database.

This request will also issue a notification in job-* vqueue with type 'update' and the following data:

```
element job {
 element job-guid { guid }
 & element account-guid { guid }
 & element device-guid { guid }
 & element update { 'state' | 'progress' } # concrete values 'state' or 'progress'
}
```

Watchdog timer on executing jobs

When the "started" element is posted on a job, a "progress" element must be posted at least every 10 minutes. Failure to post a progress element every 10 minutes will cause the backend to consider the job as hung, and will cancel the job. This will allow us to safely detect when systems reboot during a job, if the service is restarted, or if network connectivity is lost for a longer period of time.

The backend may use more than 10 minutes for the timeout - the connectors however must ensure that a new progress PUT message is transmitted no later than 10 minutes after the previous.

A note about job cancellation

The backend may choose to cancel a job at any time - also while it is executing.

When the connector PUTs the next progress indication (or a started/failed/succeeded indication), that call will fail with a status 409 (Conflict) error. This is a simple notification mechanism that will let the connector know the job was requested cancelled, without the connector needing to refresh the job description.

About job statistics

Each job also can have some stored statistics. One job can have an arbitrary number of statistic information. Each statistic record (stat) need to have name and value.

Method GET /users/{user-id}/devices/{device-id}/jobs/{job-id}/stats

This endpoint returns all stored stats for a specified job. A user should have G/DevJobStats ACL to use this endpoint.

Response

The response document will adhere to:

```
element stats {
 element stat {
  element name { text }
  & element value { text }
  }*
}
```

Method GET /users/{user-id}/devices/{device-id}/jobs/{job-id}/stats/{stat-name}

This endpoint returns stat value for a specified stat name. A user should have G/DevJobStats ACL to use this endpoint.

Response

The response document will adhere to:

```
element value { text }
```

Method PUT /users/{user-id}/devices/{device-id}/jobs/{job-id}/stats/{stat-name}

This endpoint creates stat by name. Only one stat with a specific name can be created. Using this method on already existing stat will just update stat value. A user should have U/DevJobStats ACL to use this endpoint.

Request

The request document must adhere to:

```
element value { text }
```

Method DELETE /users/{user-id}/devices/{device-id}/jobs/{job-id}/stats/{stat-name}

This endpoint delete stat by name. A user should have D/DevJobStats ACL to use this endpoint.

Logging API

Logging API covers 2 sets of endpoints for regular log messages and for audit logs.

Regular logs

Regular logs store developer defined messages and stored for a limited time. Follow this link for more information: Logging framework overview

Audit logs

Audit logs store history of actions made by Veritas SaaS Backup users. They are never cleared and stored in a special format uniquely identifying an endpoint they were generated for.

We primarily audit actions changing the state of the system, harmless GET and HEAD requests (with some exclusions) are not audited.

Since audit logs contain user facing messages they may be converted to user-readable strings which may also be translated into different languages.

Method GET/PUT /audit/filter

Retrieves raw audit logs, which in general cannot be presented to the user without further processing. This method should be used internally for investigation.

Request

The request body must adhere to the following schema

```
element filter {
 element account { text } # Account to query logs for
 & element token { text }? # Filter by token (user)
 & element recursive { boolean }? # Retrieve logs for subaccounts too
 & element from { timestamp }? # Return logs in window [from...to]
 & element to { timestamp }?
 & element allowed { boolean }? # Return only allowed actions
 & element acl { # Filter by acl (name/method pairs)
   element name { text }? # ACL name supported by the system
   & element method { text }? # HTTP method
 }*
}
```

Result

The resulting document has the following schema:

```
element audit {
 element record {
  element account { text }?
  & element device { text }?
  & element token { text } # Token which performed an audited action
  & element acl { text } # acl/method identifying an endpoint
  & element method { text }
  & element allowed { boolean } # Indicates where an action was allowed by the system
  & element time { timestamp }
  & element client-ip { text }?
  & element metadata { # metadata attached to the record
    element parameter {
     element key { text }
     & element value { text }
    }*
  }?
 }
}
```

Method GET/PUT /audit/filter/pretty

This endpoint accepts the same request body as a method above but performs translation of the technical audit record representation to a human-readable form.

Result

The resulting document has the following schema:

```
element audit {
 element record {
  element account { text }?
  & element token { text } # Token which performed an audited action
  & element message { text }
  & element acl { text } # acl identifying an endpoint
  & element allowed { boolean } # Indicates where an action was allowed by the system
  & element client-ip { text }?
  & element time { timestamp }
  }
}
```

Messaging and miscellaneous endpoints

In-app mail

The following methods provide access to the in-application mailbox associated with each user account. Whenever we wish to contact a user, we will post a mail to his in-app mailbox and optionally also send it by e-mail.

Method GET /bl/mailbox/{account-guid}

This method returns the list of non-deleted messages in the user's mailbox. Response adheres to:

```
element mailbox {
 element mail {
  element guid { text }
  & element subject { text }
  & element delivered { iso8601-time }
  & element read { iso8601-time }?
 }*
}
```

Method GET /bl/mailbox/{account-guid}/{mail-guid}

This method returns the HTML contents of the message. The response is simply a text/html document containing the message HTML document. No schema is provided.

This method will mark the message as read if it was not already marked as read.

Method DELETE /bl/mailbox/{account-guid}/{mail-guid}

This method will mark the specified message as deleted.

Time zone support

The following endpoint will retrieve the named time zones currently supported by the back-end

Method GET /bl/timezones

Response

The response adheres to:

```
element timezones {
 element zone { text }*    # for example "Europe/Copenhagen" ...
}
```

Actually, we support the "TZ" or "zoneinfo database" time zones as described in:
https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

This endpoint will return that list. The endpoint should of course be employed, as it is the authoritative list of what the back end implements.

Miscellaneous Endpoints

Method GET /download/{directory-object-hash}/{filename}

In order to download files, we provide a simple GET method which only requires the hash of the directory object, and the name of the file in the directory that the user wishes to download.

This API call is not currently authenticated. Since the knowledge of the directory object hash itself proves knowledge of the directory and all its contents, the hash alone is sufficient authentication. However, the clients should not provide the user with easy access to the link, as the link cannot easily be revoked in case the user accidentally submits the link to a third party.

In order to download entire folder as zip archive, URI should look like: /download/{directory-object-hash}/:zip. Also, it is possible to provide alternative name for zip archive, for this user needs to add header "zip-alternative-name" otherwise archive will be named just 'archive.zip'.

Response

The response will have a content-type of application/octet-stream and a content-disposition of attachment with filename set to the given name. Please note, as per RFC 2183, that the filename must consist only of US ASCII characters!

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	OK

Method POST /queue/{queue-id}

In order to create an event, one-shot or recurring, a user can issue a POST on this end point. Any queue-id can be used - the queue does not need to be created separately.

Request

The request body must adhere to this schema:

```
element event {
 element identifier { text }
 & element expire { iso8601 timestamp }?
 & element period { iso8601 period }?
 & element attributes {
   element attribute {
     element key { text }
     & element value { text }
   }*
 }?
}
```

If the expire element is not supplied, the event is scheduled to expire immediately (eg. be returned by GET on the queue).

If the period element is not supplied, the event is a one-shot event and it will be deleted from the queue the first time it is returned from GET. If period is supplied, the event gets re-scheduled whenever it is returned from GET.

Example

```
POST /queue/o365 HTTP/1.1
host: api
authorization: Basic ZmRyaHljZG94d3RlcmdqZzpwYmJzdm13dGp5Z291d3Jk
content-length: 104
<event>
 <identifier>/users/fchlqe-qty17l-v4f07g/devices/p4kaoh-rcqzxk-t1zwyb</identifier>
 <period>PT10M</period>
</event>
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Method GET /queue/{queue-id}

This will either return a 404 status code if no event has currently expired on the given queue, or it will return the oldest expired event on the queue and then either delete or re-schedulethe event (depending on whether it is a one-shot or recurring event).

We require a system user account on this API endpoint, as it affects events which are created by all users on the system.

Response

Assuming an event has expired, the event description is returned in the response body document.

```
element event {
 element identifier { text }
 & element account { text }
 & element attributes {
    element attribute {
      element key { text }
      & element value { text }
    }*
 }
}
```

The account element will hold the account identifier for the account that originally asked the event to be scheduled. This account identifier can be used in subsequent requests in the "impersonate" header, to impersonate the given user when working on whatever object was scheduled for treatment in the event.

Attributes array will contain a set of Key-Value pairs that will store arbitrary data (w/o fixed format).

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	FAIL	FAIL	OK	FAIL

Method DELETE /queue/{queue-id}/{event-identifier}
This is used to delete an event from a queue. The identifier used when describing the element in the POST and GET documents is URL-encoded when used in the URI for this endpoint.

Example
```
DELETE /queue/o365/%2Fusers%2Ffchlqe-qty17l-v4f07g%2Fdevices%2Fp4kaoh-rcqzxk-t1zwyb
HTTP/1.1
```

The request should be authenticated as the user that created the event.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	FAIL	FAIL	FAIL

Auxilary endpoints

Method GET /time
This method is used to request API server's time

Response
```
element time { iso8601 timestamp }
```

Calling of this endpoint is cheap and therefore may be used by desktop clients to check that server is alive.

Method GET /o365
Retrieve information about Office 365 sub accounts for currently authenticated user.

Used by old versions of clients and should not be used anymore in the new clients. Endpoint '/o365/users' should be used instead now (see below).

Response
```
element subaccounts {
 element subaccount {
  element email { text }
  & element title { text }
 }*
}
```

Method GET /o365/users
Retrieve information about Office 365 sub accounts for currently authenticated user. Request header 'x-o365-device' should be supplied and contain id of o365 device.

Response
```
element subaccounts {
 element subaccount {
  element id { text }
  & element email { text } ?
  & element title { text } ?
  & element userPrincipalName { text } ?
  & element userType { text } ?
  & element hasSharepointAccess { text } ?
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	OK	FAIL	FAIL

Method GET /o365/users/{userPrincipalName}

Retrieve information about Office 365 account by userPrincipalName. Request header 'x-o365-device' should be supplied and contain id of o365 device. We can set userPrincipalName to "me" to get information about current admin user.

Response

```
element account {
 element id { text }
 & element email { text } ?
 & element title { text } ?
 & element userPrincipalName { text } ?
 & element userType { text } ?
 & element hasSharepointAccess { text } ?
}
```

Method GET /o365/groups

Retrieve information about Office 365 groups for currently authenticated user. Request header 'x-o365-device' should be supplied and contain id of o365 device.

Response

```
element groups {
 element group {
  element id { text }
  & element email { text }?
  & element title { text } ?
  & element description { text }?
  & element securityEnabled { boolean }
  & element mailEnabled { boolean }
  & element groupTypes {
   & element groupType { text }*
  }
  & element visibility { text }?
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	OK	FAIL	FAIL

Method GET /o365/exp_groups

Retrieve information about Office 365 groups and their members for currently authenticated user. Request header 'x-o365-device' should be supplied and contain o365 device id.

Response

```
element groups {
 element group {
  element id { text }
  & element email { text }?
  & element title { text } ?
  & element description { text }?
  & element securityEnabled { boolean }
  & element mailEnabled { boolean }
  & element groupTypes {
   & element groupType { text }*
  }
  & element visibility { text } ?
  & element members {
   element member {
    element id { text }
    & element email { text } ?
    & element title { text } ?
    & element userPrincipalName { text } ?
    & element userType { text } ?
    & element hasSharepointAccess { text } ?
   }*
  }
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	OK	FAIL	FAIL

Method GET /o365/groups/{group-id}/members

Retrieve information about members of group specified by group-id. Request header 'x-o365-device' should be suplied and contain o365 device id.

Response

```
element members {
 element member {
  element id { text }
  & element email { text } ?
  & element title { text } ?
  & element userPrincipalName { text } ?
  & element userType { text } ?
  & element hasSharepointAccess { text } ?
 }*
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	FAIL	OK	OK	FAIL	FAIL

Method GET /o365/site_collections

Retrieve information about Office 365 sharepoint site collections. Request header 'x-o365-device' must be supplied and contain id of o365 device.

Response

```
element site_collections {
 element collection { text } # URI of site collection
}
```

ACL

G/O365SiteCollections ACL must be enabled for the account to be able to access this endpoint.

Method GET /o365/sites

Retrieve information about Office 365 sharepoint top level sites. Request header 'x-o365-device' must be supplied and contain the id of o365 device. Return 503 error if Office 365 server return 5XX error.

Response

```
element sites {
 element site {
  & element url { text }
  & element displayName { text }
 }*
}
```

ACL

G/O365Sites ACL must be enabled for the account to be able to access this endpoint.

Method GET /o365/sites/{masked-site-url}/sites

Retrieve information about Office 365 sharepoint top level sites. Request header 'x-o365-device' must be supplied and contain the id of o365 device. Return 503 error if Office 365 server return 5XX error. Return 423 error if subsite is blocked. Return 404 error if provided path does not exist. Please note that site URL should be masked.

Response

```
element sites {
 element site {
  element id { text }
```

```
   & element url { text }
   & element name { text }
   & element displayName { text }
  }*
 }
```

ACL

G/O365SubSites ACL must be enabled for the account to be able to access this endpoint.

Method GET /o365/clientid

Obtain environment dependent Office365 admin app OAuth client id.

We now support multiple apps for different customer domains on a single environment. Set client_domain attribute on the account to get correct one. Currently supported values are "default" and "veritas". Missing attribute is equivalent to "default" value.

This endpoint supports 2 attributes telling which application id to return:

- x-o365-environment - may have 2 values: default (global application set) or germany (applications for German Office365). Missing header is equivalent to default
- x-app-version - must contain concrete app version. i.e. 1.0, 2.0, 3.0, etc

Response

```
element id { text }
```

ACL

G/O365ClientId ACL must be enabled for the account to be able to access this endpoint.

Method GET /gsuite/users

Retrieve information about GSuite sub accounts for currently authenticated user. Request header 'x-gsuite-device' and contain id of gsuite device and 'x-gsuite-domain' containing target domain should be supplied.

Response

```
element subaccounts {
 element subaccount {
  element id { text }
  & element email { text }
  & element title { text }
 }*
}
```

ACL

G/GSuiteUsers ACL must be enabled for the account to be able to access this endpoint.

Method GET /gsuite/domains

Retrieve information about GSuite sub accounts for currently authenticated user. Request header 'x-gsuite-device' and contain id of gsuite device should be supplied.

Response

```
element domains {
 element domain {
  element name { text }
 }*
}
```

ACL

G/GSuiteDomains ACL must be enabled for the account to be able to access this endpoint.

Method GET /gsuite/clientid

Obtain environment dependent GSuite app OAuth client id.

We now support multiple apps for different customer domains on a single environment. Set client_domain attribute on the account to get correct one. Currently supported values are "default" and "veritas". Missing attribute is equivalent to "default" value.

Response

```
element id { text }
```

ACL

G/GSuiteClientId ACL must be enabled for the account to be able to access this endpoint.

Method GET /salesforce/clientid

Obtain environment-dependent Salesforce app OAuth client id.

We now support multiple apps for different customer domains on a single environment. Set client_domain attribute on the account to get correct one. Currently supported values are "default" and "veritas". Missing attribute is equivalent to "default" value.

Response

```
element id { text }
```

ACL

G/SalesforceClientId ACL must be enabled for the account to be able to access this endpoint.

Method GET /salesforce/users/count

Retrieve information about Salesforce device users counts for the currently authenticated user. Request header 'x-salesforce-device' containing the id of salesforce device should be supplied.

Response

```
element count { integer }
```

ACL

G/SalesforceUsersCount ACL must be enabled for the account to be able to access this endpoint.

Object Encoding

Version 0 objects

Version 0 was used in the original backup library from 2013. The main weakness of the V0 format is, that it supports a fixed set of objects (windows and linux files and directories) and each object type must have one certain set of properties.

The objects consist of two parts; a header section and a data section. The header section is required to adhere to a specific format for the back end systems to be able to manage the data appropriately - for example in order for data expiry to function.

The data section is more flexible. This section is not used by the back end systems, so the contents are entirely up to the clients (KNG/API users). However, in order to provide a consistent and interoperable product, we want the clients to agree on exactly how we represent directories, files, links, and other file system objects as well as their vast range of operating system specific attributes such as ownership, permissions, ACLs, MAC label, etc.

Basic element encoding

We use 1, 2, 4 and 8 byte unsigned integers for the encoding of almost all data in the objects. These integers are encoded in network byte order (big endian).

We also use text strings - a text string is a 4-byte integer (denoting the length in bytes of the text) followed by the UTF-8 text string. There is no termination marker (like zero termination or other) on the string - the string takes up the 4 bytes for the length followed by the exact number of bytes as given in the length specifier.

Quota use

All referrer objects must have the sum of the size of their sub-trees plus their own size stored in their meta data section. The server will validate this number on POST. The number will be a 64-bit unsigned integer.

This also means that we cannot create an object with a reference to an object that does not exist, because then we cannot validate the quota accounting.

Common header for all objects

All objects stored on our back end will hold a version number identifier as the first byte. This will allow us to upgrade our storage model in the future.

After that, an object type identifier is stored. This type is currently only needed for extra validation - when traversing object hierarchies it will be evident which type a given objects should be, based on its parent (for example, a file entry will refer to file data object types, a directory entry will refer to a directory data object type).

Offset	Length	Description	Value
0	1	Version	0
1	1	Object type	see below

Aside from these header bytes, different object types will have different header content.

Common encoding of List-of-References: The LoR

Any object that refers to other objects must contain the hashes of those referred objects in a LoR, a List of References, in the header of the object. This will be required for the server in order to properly garbage collect data that is no longer referenced by any user accounts backup history or archive.

The LoR is a list of "multi-references". Each multi-reference will refer to zero or more objects, and is therefore in itself a list. Each reference in the multiref is a simple fixed length 32-byte raw hash value.

So, in other words, the LoR is a list of lists of 32-byte blocks.

Every list is constructed as a four-byte element counter followed by the elements.

The LoR is constructed as such:

Offset	Length	Description	Value
0	4	Number of multirefs	n
4	4	Refs in MR0	r0
8	r0 * 32	hashes in MR0	
	4	Refs in MR1	r1
	r1 * 32	hashes in MR1	
		…	

Each 32 byte hash can be translated directly to textual representation (for use in API requests) simply by converting byte-for-byte from 1-byte integer to 2-character lower case hexadecimal.

General "directory" object

Our directory object supports directories on any platform (so far). Its data section will contain platform specific encodings or directory child object meta data - such as a Linux pipe or an NTFS file.

Offset	Length	Description	Value
0	1	Version	0
1	1	Object type	0xdd or 0xde
2	8	Tree-size	…
10	…	LoR	
…	…	LoM	

Object type 0xdd is used to mark "partial" directories, i.e. the ones whose subdirectories do not reflect user's folder structure to full extent. They are considered to be temporary and used mainly to give user an impression of that files appear and accessible on server in realtime, even if initial backup in not completed yet.

Object type 0xde is normal ("complete") directory entry.

The Tree-size is the sum of the size of the current object as encoded on the back-end and the Tree-size of all its child objects (referred in the LoR).

The "LoM" is the List-of-Metadata. The LoM contains exactly the same number of entries as the LoR, and therefore we do NOT have an integer for specifying the number of elements in the LoM.

LoM entry: UNIX regular file

When we find a LoM entry for a UNIX regular file, the corresponding LoR entry will either have zero hashes (if the file is empty), or one or more hashes. In order to limit the size of objects we upload, large files will be split into multiple objects. The objects referred to in the LoR will be "file data" objects (type 0xfd).

Offset	Length	Description	Value
0	1	Meta Type	0x01
1	...	Name	string with file name
...	...	User-name	string with owner name
...	...	Group-name	string with group name
	4	Permissions	UNIX permissions & 07777
	8	Change-time	ctime
	8	Modification time	mtime
	8	File size in bytes	...

LoM entry: UNIX directory

When we find a LoM entry for a UNIX directory, the corresponding LoR entry will have one or more hashes. If more than one hash is encountered, we must process the referred objects in turn, combining their content - it is simply a very large directory that has been split into multiple objects to limit the maximum object size. Thus, the objects referred to in the LoR will be "directory entry" objects (types 0xdd or 0xde).

Offset	Length	Description	Value
0	1	Meta Type	0x02
1	...	Name	string with file name
...	...	User-name	string with owner name

Offset	Length	Description	Value
...	...	Group-name	string with group name
	4	Permissions	UNIX permissions & 07777
	8	Change-time	ctime
	8	Modification time	mtime

LoM entry: Windows regular file

When we find a LoM entry for a Windows regular file, the corresponding LoR entry will either have zero hashes (if the file is empty), or one or more hashes. In order to limit the size of objects we upload, large files will be split into multiple objects. The objects referred to in the LoR will be "file data" objects (type 0xfd).

Offset	Length	Description	Value
0	1	Meta Type	0x11
1	...	Name	string with file name
...	...	User-name	string with owner name
...	...	SDDL	string with SDDL
	4	file attributes	refer to http://msdn.microsoft.com/en-us/library/windows/desktop/aa365535%28v=vs.85%29.aspx
	8	Modification time	ctime - last write time
	8	Birth time	btime - time the file was created
	8	File size in bytes	...

LoM entry: Windows directory

When we find a LoM entry for a Windows directory, the corresponding LoR entry will have one or more hashes. If more than one hash is encountered, we must process the referred objects in turn, combining their content - it is simply a very large directory that has been split into multiple objects to limit the maximum object size. Thus, the objects referred to in the LoR will be "directory entry" objects (types 0xdd or 0xde).

Offset	Length	Description	Value
0	1	Meta Type	0x12

Offset	Length	Description	Value
1	...	Name	string with file name
...	...	User-name	string with owner name
...	...	SDDL	string with SDDL
	4	file attributes	refer to http://msdn.microsoft.com/en-us/library/windows/desktop/aa365535%28v=vs.85%29.aspx
	8	Modification time	ctime - last write time
	8	Birth time	btime - time the file was created

General "file data" object

Since file meta data is kept in the directory entry object, we should be able to use a single object type for file content data from all types of files from various operating systems. This object type is the general "file data" object intended for this use. The following is the header for this object type:

Offset	Length	Description	Value
0	1	Version	0
1	1	Object type	0xfd

After this header follows the actual file data.

There is no LoR, so a file data object cannot refer to other objects.

Version 1 objects

The V1 format was introduced to be used with synchronization. It should eventually replace V0 in every part of our system.

Overview

The V1 format supports two main types of objects: Containers and Leaves. The Container holds a list of "pointers" to other objects (which can be Containers or Leaves), as well as a set of properties for each of those objects. The Containers can thus be used to build up a directory hierarchy as known from a normal PC file system, or it can be used to hold mails in a mailbox or other things we may think of in the future - the Container is fairly general in this respect. The Leaf is an object that holds data and does not refer to other objects - a Leaf will usually be a file on a computer, a symlink, a mail, etc.

Object slicing

No object may grow larger than 8MiB+2B (8388610 bytes). In case we need to encode file data or directory contents (or anything else for that matter), and the Container or Leaf object that we are encoding grows above this limit, we will "slice" the object. What this means, is, we encode multiple objects (complete with version and type specifier in each object), and refer to the collective object with a sequence of hashes instead of a single hash.

We refer to an object with its hash - a fixed 256 bit value. If an object is sliced into two, we will refer to the full object with a 512 bit value. The object is decoded simply by reading each of the slices in order, decoding the slice and adding the decoded results to the in-memory object we are decoding (or writing to the on-disk file we are restoring).

The Container

The container is encoded the following way:

Element	Type	Description
Version	uint8_t	0x01 - the object version
Object Type	uint8_t	0x00 - the Container designation
Children...	ChildRef	sequence of ChildRef

The ChildRef is an encoding of a reference to a child - it is done this way:

Element	Type	Description
ChildType	uin8_t	0x00 for Container, 0x01 for Leaf
Name	UTF8 string	Name of child (must be unique in this Container)
Size	uint64_t	Tree-size of child
Hash	sha256*	Hash sequence - 0 or more hashes
Properties	Property*	Property sequence - 0 or more properties
End	uint8_t	0x00

The Property is an encoding of a named property we associate with the child, aside from the mandatory properties (Name, Size and HashSeq).

Element	Type	Description
Type	uint8_t	String=0x05, HashSeq=0x04, uint64_t=0x03, uint32_t=0x02, no-value=0x01
Name	UTF8 string	Name of property
Value	Value?	specific encoding of typed value - nothing if no-value type 0x01

The Leaf

The leaf is just a lump of data - it has meaning only by virtue of the Container that references it. For example, file data has no name - it is the Container which names the file and refers to its data. The encoding of the Leaf is as follows:

Element	Type	Description
Version	uint8_t	0x01 - the object version
Type	uint8_t	0x01 - the Leaf designation
Payload	bytes...	the payload data

Currently used dynamic property names:

Name	Type	Meaning
deleted	no-value	Sync code will mark files as being deleted but not actually remove them from the Container - thus, an entry with this property set should be hidden and considered "not there". This is only used by the sync code (on-line drives).

Performance Logging API

The performance logging API allows centralized storage of performance metrics for high-frequency events or high-volume data; such as query processing times or object sizes.

Overall workings

Each "client" (be it an internal server or an end-user device running our software) will have its code instrumented certain places. Those instruments will collect and aggregate high-frequency or high volume data, and periodically report back those aggregates.

Each "meter" is defined by a client name and a meter name.

The configuration of a meter covers the unit of the quality measured (seconds, bytes, ...), and the scale on which we represent it (eg. "log0" which is our first log scale and currently the only defined scale).

Each aggregate data record contains the start time, the period it spans, and a "data vector". The data vector is a space separated list of decimal integers; each pair if integers is a bucket number and a bucket count. Only buckets with non-zero counts are reported.

This allows for efficient aggregation, transmission, storage and querying of histogram data.

Scales in use

Any measured quantity needs to be accounted against some bucket number. The scale defines this conversion. These are the defined scales:

Name	Definition	Explanation
opg	Occurrence per string group	Can be used to count occurrence of event per group. Groups can be represented as strings. E.g. number of requests per IP address
log	Logarithmic scale	min - low bound of reported value max - high bound of reported value len - number of buckets Intended for time or size measurements First bucket also holds values are less then low bound Last bucket also holds values are greater then upper bound.
opgi	Occurrence per int group	The idea is the same as OPG scale but has another visualization that help monitoring a value over a time.

The "data vector"

When meters send data to the API or when clients receive performance data from the API, data is encoded in a "data vector". This is a compact yet human-readable representation of the histogram bucket data. It is a space-separated list of decimal integers; each pair of integers is the bucket number and the event count for that bucket.

Thus, an event meter using the logo scale which has recorded the following events:

```
1ms
1ms
10ms
100ms
2ms
5ms
```

Would report the following data vector:

```
375 2 412 1 462 1 500 1 625 1
```

Method GET /evperf/{peer-id}/{meter-id}

Returns the configuration of the meter if it exists, 404 otherwise.

The peer-id will be the hostname if it is an internal host, or device-guid if it is an end-user device.

The configuration document adheres to the following schema

```
element config {
  element unit { text }
  & element scale { text }
  & element info { text }?
}
```

Method POST /evperf/{peer-id}

Create a new meter, given some meter configuration

The document must adhere to the following schema

```
element meter {
 element name { text }
 & element unit { text }
 & element scale { text }
 & element info { text }?
}
```

Method POST /evperf/{peer-id}/{meter-name}

Send event performance data. The request document must adhere to the following schema

```
element data {
 element start { timestamp }
 & element span { period }
 & element data { text }
}
```

The data field is a "data vector" as per the format description above.

Method GET|PUT /evperf/{peer-id}/{meter-name}/query

Query performance data.

The request document must adhere to the following schema

```
element query {
 element start { timestamp }
 & element span { period }
}
```

The result will be a text response holding a "data vector" as described above.

Search API

BSearch interface

The BSearch interface is an alternative to the Search interface. It uses a "/bsearch" endpoint rather than "/search":

```
curl -v -k -XGET 'https://joe:*****@localhost:8443/users/add227e7-7f01-4641-803a-
864a920aa816/bsearch' --data-urlencode 'searchTerms=hello world' --data-urlencode
'count=3'
```

Aside from that, there are smaller differences in the API - those will be summarized here in this section.

The query terms

BSearch supports basically the same set of query parameters as basic Search, plus an extra parameter "pathRoot" that is useful when using BSearch to provide rich meta-data to a browsing experience:

Parameter	Explanation
searchTerms	Space separated list of terms to search for (if not given, all objects will match). This is fuzzy search. Support of the exact search has been terminated.
itemName	Exact name of the item to search for. It should not be used for classical search but rather for such cases as retrieving history of changes for a given item.
pathRoot	Full path to the container of files we wish to consider in the search (if not provided, search will search across all objects regardless of location). At least pathRoot or searchTerms must be specified!
device	If pathRoot was supplied, a device guid must also be specified (to give the path its necessary context)
sortTerms	Comma separated list of sort terms. Valid terms are "title" to sort on the object title (object name), or any meta-data key name prefixed with "meta:". For example, sorting by "meta:Subject" will sort by e-mail subject.
count	Maximum number of results in response. If not given, 100 is used.
startIndex	Index of first result. If not given, 0 is used.
snaptime	Disregard any information that appeared after this ISO8601 timestamp (in other words, search/browse device(s) as it/they were at the given time)
medium	The type of objects to match (based on detected mime type of data). Valid values can be one of: **image, audio, video, document**
snaptimeFrom	Define snapshot range (timestamp from)
snaptimeTo	Define snapshot range (timestamp to)
deleted	Include deleted files (default is false)
fullHistory	Return full history of requested item (recommended to use with searchTerms="filename")
includeBody	Return full response (default is true). Should be used to get totalCount only
filters	Specify object types to be returned (same as medium + deleted, sys…).
sys	Include sys files to response. Default is false
recursive	Enable recursive search in specified pathRoot

Sorting terms

In sortTerms the following entries are recognized:

Parameter	Meaning
isfolder	Folders first, non-folders follow
isreceived	If meta:received property exists item is first, others follow
title	meta:subject if exists, otherwise name of object
name	name of object
updated	Time of snapshot in which object was last updated
time	Extracted timestamp from meta:received if present, otherwise from meta:date if present, otherwise from mtimes if present, otherwise time of snapshot in which object was updated
error	When searching, error=0 means perfect match, error increases as match quality deteriorates
meta:XXX	Lexicographical sort by meta-data item XXX

Any parameter can be prefixed with '-', which will cause the sorting order to be reversed.

Any number of parameters can be given, separated by commas. Valid examples include: "isfolder,title" and "isfolder,isreceived,-time".

The response document:
BSearch also returns an Atom document like opensearch. BSsearch extends the response entry with an additional namespace for Veritas SaaS Backup specific data (such as supplemental object metadata). An example response document could look like:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/" xmlns:media="http://search.yahoo.com/mrss/"
xmlns:kng="http://saasbackup.veritas.com/opensearch/1">
  <opensearch:itemsPerPage>100</opensearch:itemsPerPage>
  <opensearch:startIndex>0</opensearch:startIndex>
  <opensearch:totalResults>5</opensearch:totalResults>
  <entry>
    <media:category>unknown</media:category>
    <id>kng://bsearch/folder0/</id>
    <title>folder0/</title>
    <media:title>/folder0/</media:title>
    <kng:name>folder0/</kng:name>
    <updated>2016-05-27T09:27:56Z</updated>
    <kng:meta key="size">70</kng:meta>
    <media:content type="folder"
url="/download/e8d96ce7ab426cd074896a76a2a29bb9a5007e19ac7dff17c38ed0ce8d5f8eee/folder0
"/>
  </entry>
  <entry>
    <media:category>unknown</media:category>
    <id>kng://bsearch/folder1/</id>
    <title>folder1/</title>
    <media:title>/folder1/</media:title>
    <kng:name>folder1/</kng:name>
    <updated>2016-05-27T09:27:56Z</updated>
    <kng:meta key="size">70</kng:meta>
    <media:content type="folder"
url="/download/e8d96ce7ab426cd074896a76a2a29bb9a5007e19ac7dff17c38ed0ce8d5f8eee/folder1
"/>
  </entry>
  <entry>
    <media:category>document</media:category>
    <id>kng://bsearch/file0.txt</id>
    <title>file0.txt</title>
    <media:title>/file0.txt</media:title>
    <kng:name>file0.txt</kng:name>
    <updated>2016-05-27T09:27:56Z</updated>
    <kng:meta key="size">10</kng:meta>
    <kng:meta key="mime-type">text/plain</kng:meta>
    <media:content medium="document" type="text/plain" fileSize="10"
url="/download/e8d96ce7ab426cd074896a76a2a29bb9a5007e19ac7dff17c38ed0ce8d5f8eee/file0.t
xt" error="0"/>
  </entry>
  <entry>
    <media:category>document</media:category>
    <id>kng://bsearch/file1.txt</id>
    <title>file1.txt</title>
    <media:title>/file1.txt</media:title>
    <kng:name>file1.txt</kng:name>
    <updated>2016-05-27T09:27:56Z</updated>
    <kng:meta key="size">10</kng:meta>
    <kng:meta key="mime-type">text/plain</kng:meta>
    <media:content medium="document" type="text/plain" fileSize="10"
url="/download/e8d96ce7ab426cd074896a76a2a29bb9a5007e19ac7dff17c38ed0ce8d5f8eee/file1.t
xt" error="0"/>
  </entry>
  <entry>
    <media:category>unknown</media:category>
    <id>kng://bsearch/file1.eml</id>
    <title>some stuff</title>
    <media:title>/file1.eml</media:title>
    <kng:name>file1.eml</kng:name>
    <updated>2016-05-27T09:27:56Z</updated>
    <kng:meta key="size">53</kng:meta>
    <kng:meta key="mime-type">message/rfc822</kng:meta>
    <kng:meta key="From">there</kng:meta>
    <kng:meta key="To">here</kng:meta>
    <kng:meta key="Subject">some stuff</kng:meta>
    <media:content medium="unknown" type="message/rfc822" fileSize="53"
url="/download/e8d96ce7ab426cd074896a76a2a29bb9a5007e19ac7dff17c38ed0ce8d5f8eee/file1.e
ml" error="0"/>
  </entry>
</feed>
<!-- End of document -->
```

Sharing API

About sharing

The purpose of the sharing API is to facilitate a simple sharing of "any backup of a given file or folder hierarchy" from an authenticated Veritas SaaS Backup user to the public.

When a user chooses to share a file or a folder hierarchy, the system will generate a link for the user. This link will be on the form:

Single file sharing	https://saasbackup.veritas.com/share/{guid}/{filename}
Folder hierarchy sharing	https://saasbackup.veritas.com/share/{guid}/

The URI that is generated when sharing a folder hierarchy, will present a web UI page that is similar to the backup browser for normal authenticated users. It will not allow device or snapshot selection, but it will allow browsing, previewing and downloading of files in the shared folder hierarchy.

A sharing URI will automatically expire after a set period of time - for example five days.

If the sharing URI points to the most recent snapshot, it is possible for shared objects to disappear from the share. The UI will have to deal with such situations gracefully. If URI points to specific snapshot this snapshot can't be deleted and its objects never disappear.

/share/ versus /s/

It is possible to use /s/ instead of /share/ in all of the following API endpoints. The /share/ component is kept for compatibility reasons.

Method POST /share/

This API method will generate a share for a given resource for the authenticated user. The location header will contain the full URI to the share. Following that URI will either cause an immediate file download (in case the shared object is a file) or present the user with a web UI for browsing (in case the shared object is a directory).

Request

The request body must adhere to the following schema:

```
element share {
 element device { text } # device guid
 & element path { text }
 & (element expires { timestamp }
    | element lifetime { iso8601-period })?
 & element snapshot { text }? # id of specific snapshot
 & element password { text }? # password needed to access the share
 & element guid { text }? # guid of created public link
}
```

If no expiration timestamp and lifetime are given, a sensible default will be used (21 days perhaps).

Timestamp should be in $ISO8601$ extended format for combined date and time. Examples follow:

```
<YYYY-MM-DD>{T, }<hh:mm:ss>[{,,.}<fractional seconds>]{Z,+00}
2013-02-05T20:41:46.123Z   (2012 February 5th, 20:41:46 UTC)
2013-02-05 20:41:46+00  (2012 February 5th, 20:41:46 UTC)
2013-02-05 20:41:46,123+00  (2012 February 5th, 20:41:46 UTC)
```

Lifetime should be in ISO8601-period format. Examples - "P21DT" (21 day); "PT1M" (1 min.).

Note: expires parameter is deprecated and shouldn't be used, use lifetime instead.

The path is a list of strings separated by "/" (slash) as the directory separator. The path must start on a slash. If the object shared is a directory (hierarchy), the path must end on a slash. If the object shared is a file, the path must - obviously - end with the name of the file.

If snapshot isn't provided - share will point to the latest one.

A client can set their own guid which will be used instead of back-end generated. This guid should be long, cryptographically strong and ideally never collide with other.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method PUT /share/{guid}

This API method allows updating the properties set on a particular share.

Request

The request body must adhere to the following schema:

```
element share {
 element device { text }? # device guid
 & element path { text }?
 & (element expires { timestamp }
    | element lifetime { iso8601-period })?
 & element snapshot { text }? # id of specific snapshot
 & element password { text }?
}
```

None of the elements are required of course, but whichever elements are provided will be updated on the existing share.

Note: To clear the password send empty <password></password> element.

Please note that it is probably not a good idea to change the device or path of an existing share (since that share URI may have been sent to users already, and they would then gain access to the new device/path through the old URI). This method is most useful for changing the expiration time of a share.

Note: expires parameter is deprecated and shouldn't be used, use lifetime instead.

To change share snapshot device guid must be specified. To set snapshot to empty send empty <snapshot></snapshot> element.

The account authenticated in the request headers must be the owner account of the share.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /share/

This API method will return a document listing all shares for the currently authenticated user.

```
element shares {
 element share {
  element guid { text }
  & element path { text }
  & element expires { timestamp }
  & element device { text } # device guid
  & element snapshot { text }?
  & element password { text }?
  & element dname { text }? # well readable alias for the share
 }*
}
```

Share points to the latest snapshot if "snapshot" is absent.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /users/{user-id}/share/

This API method will return a document listing all shares for the user supplied in the URL.

This method basically duplicates functionality of GET /share, but we need it to list shares of sub users in admin panel.

```
element shares {
 element share {
  element guid { text }
  & element path { text }
  & element expires { timestamp }
```

```
   & element device { text } # device guid
   & element snapshot { text }?
   & element password { text }?
   & element dname { text }? # well readable alias for the share
  }*
}
```

Share points to the latest snapshot if "snapshot" is absent.

Note: Password is only exposed to the owner of the share. Other users will receive empty <password> element meaning this share is password protected.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method DELETE /share/{guid}

This API method will delete the given share, if (and only if) the share is owned by the currently authenticated user.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	FAIL

Method GET /share/{guid}

This API method requires no authentication. It is typically executed by a normal user agent (a browser).

This method will respond with the index document of the /share/ subdirectory under the static file root directory. This index document should present a file browser to the user.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	OK	OK	OK	OK	FAIL	OK

Method GET /share/{guid}/root/{path}

Note: This endpoint is also reachable as: /share/{guid}/{path}.

This API method requires no authentication. It is executed by the web UI when the user is browsing a shared directory hierarchy. When the user types "https://saasbackup.veritas.com/share/{guid}" in his browser, he will be presented with a browsing UI. The UI will immediately execute a GET on "https://saasbackup.veritas.com/share/{guid}/root" in order to get the contents of the root of the shared hierarchy.

The API server maps the supplied path to the actual directory or file objects in the most recent snapshot on the device from which data was shared.

If share is protected with password it must be supplied in request header "share-password" as base64 encoded value. If password is wrong or not supplied 403 code and error document is returned:

```
element error {
 element code { text }
}
```

Same 403 code and document is returned if user has no rights to access the share.

Code may have the next value:

- AuthenticationRequired
- PasswordRequired
- InsufficientPermissions

In case the path leads to a file, the response will be the file data with a content disposition header set just like the regular file download API. In other words, when the path points to a file, the file is downloaded.

In case the path leads to a directory, the response will be an XML document describing the content of the directory. That response will adhere to the following schema:

```
element directory {
 element alias { text }? # gets expired after some time. Call this method again to
renew it.
 & element name { text }
 & element owner { text }
 & element treesize { integer }
 & element dname { text }? # well readable directory alias read from 'dname' property
 & (element file {
     element name { text }
     & element subject { text }?
     & element modified { text }?  # not all connectors set modified time
     & element size { integer }
     & element dname { text }? # well readable file alias read from 'dname' property
    }
    | element directory {
       element name { text }
       & element folder_type { integer }?  # it is used to display different icon for
such directories
       & element treesize { integer }?  # we may or may not have size info on
directories
       & element dname { text }? # well readable directory alias read from 'dname'
property
      }
    )*
}
```

In order to download all content of shared folder as zip archive, URI should look like: /share/{guid}/root/:zip. Also it is possible to provide alternative name for zip archive, for this user needs to add header "zip-alternative-name" otherwise, archive will be named just 'archive.zip'.

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	OK

Method GET /share/{guid}/{filename}/meta

This API method requires no authentication. It provides meta information about single shared file in the most recent snapshot on the device from which data was shared.

Response

The response will adhere to the following schema:

```
element file {
 element alias { text } # gets expired after some time. Call this method again to renew
it.
 & element name { text }
 & element owner { text } # owner guid
 & element subject { text }? # well readable alias for the share
 & element size { integer }
 }
```

If share is protected with password it must be supplied in request header "share-password" as base64 encoded value. If password is wrong or not supplied 403 code and error document is returned:

```
element error {
 element code { text }
}
```

Same 403 code and document is returned if user has no rights to access the share.

Code may have the next value:

- AuthenticationRequired
- PasswordRequired
- InsufficientPermissions

ACL

Public	Partner	Admin	User	Device	System	NONE
Ok	OK	OK	OK	OK	OK	OK

Single sign on API

Overview

Single sign on (SSO) allows users to configure their own identity provider (IdP) which Veritas servers will trust instead of checking user credentials on its own. This makes it possible for users to have the only password (for example AD password) to access Veritas SaaS Backup and other service providers (SP) supporting SSO.

Configuration of identity provider

Veritas SaaS Backup uses SAML2 with the HTTP Redirect binding for SP to IdP and expects the HTTP Post binding for IdP to SP. Veritas requires the SAML response to be signed. SAML assertions are not signed.

Identity provider must provide user's email address as NameID in "urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress" format.

Veritas SaaS Backup post back URL depends on the datacenter location:

- EU: https://eu.saasbackup.veritas.com/sso/login
- US: https://us.saasbackup.veritas.com/sso/login
- AP: https://ap.saasbackup.veritas.com/sso/login

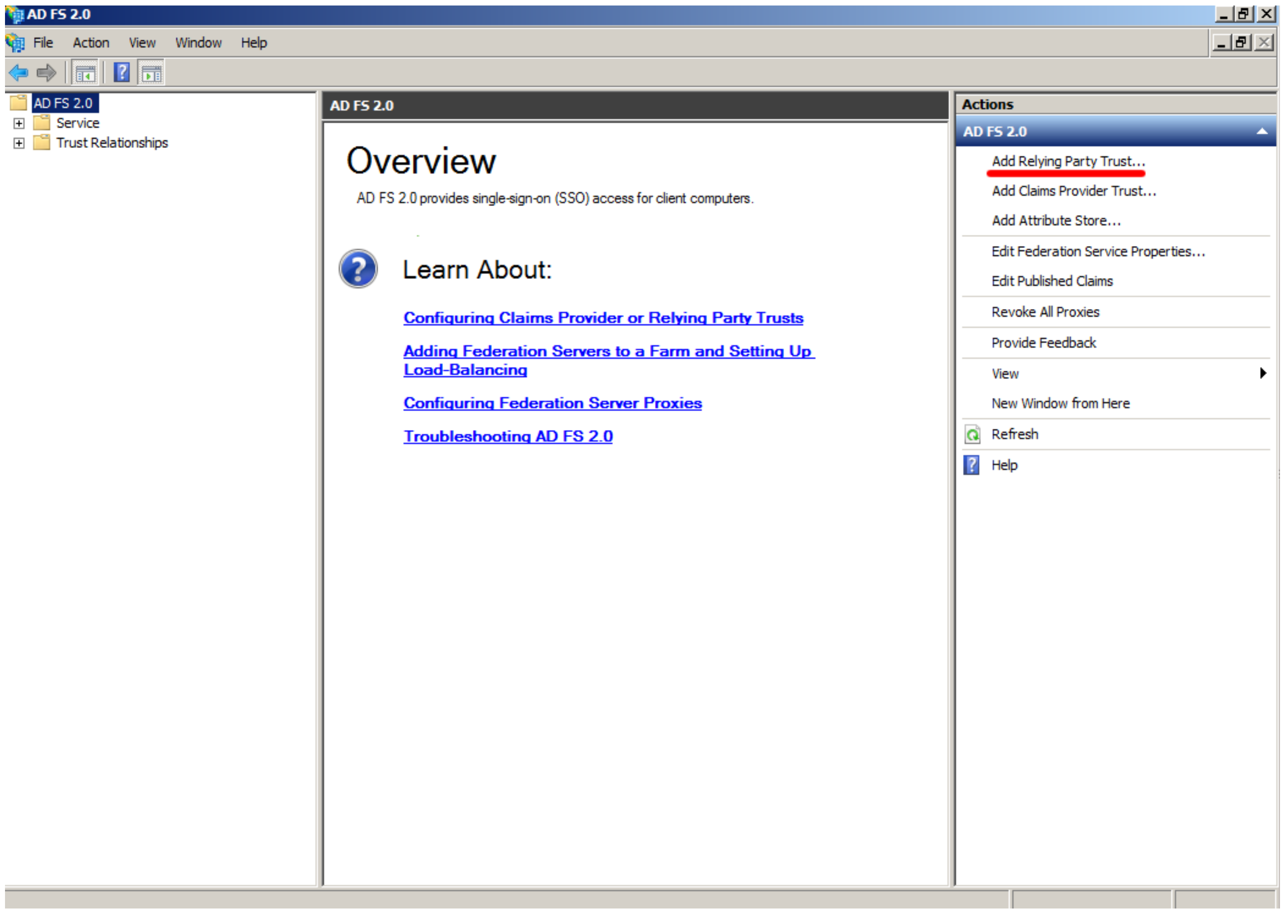
and may be extracted from SSO metadata URL

- EU: https://eu.saasbackup.veritas.com/sso/metadata
- US: https://us.saasbackup.veritas.com/sso/metadata
- AP: https://ap.saasbackup.veritas.com/sso/metadata

AD FS 2.0 sample configuration

Open AD FS 2.0 management console and add new relying party trust.

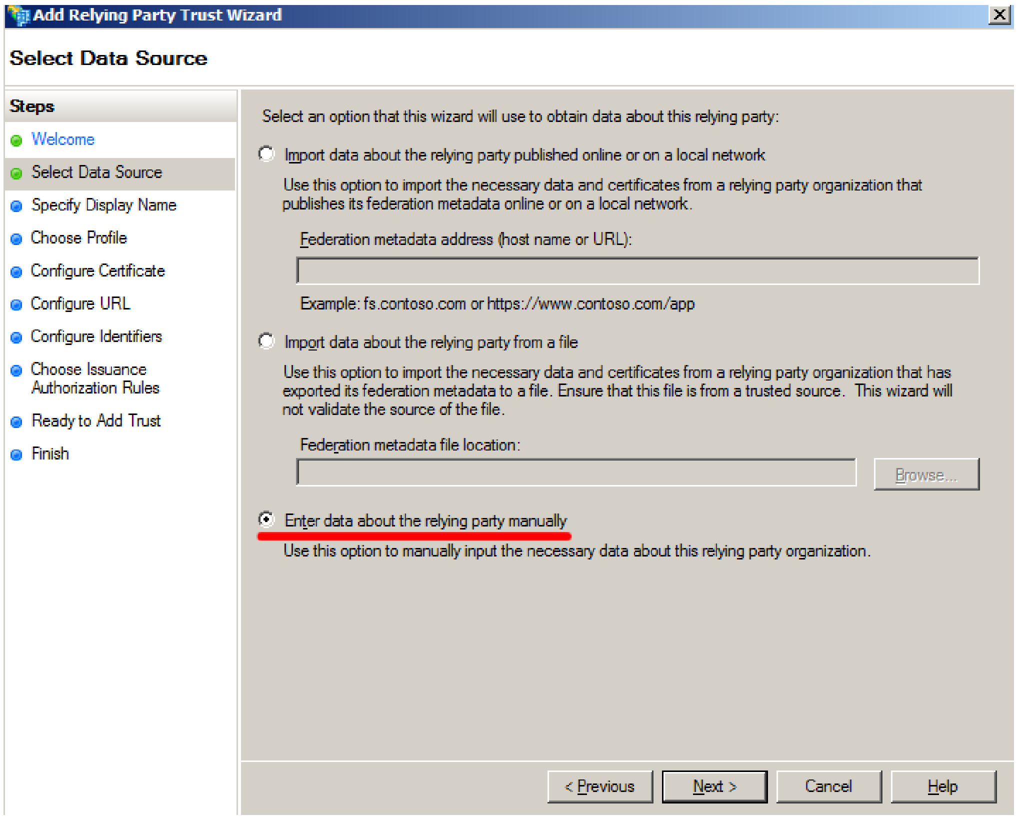
1. Open AD FS 2.0 management console and add new relying party trust.



AD FS 2.0 provides single-sign-on (SSO) access for client computers.

Configuring Claims Provider or Relying Party Trusts
Adding Federation Servers to a Farm and Setting Up Load-Balancing
Configuring Federation Server Proxies
Troubleshooting AD FS 2.0

2. Choose manual relying party configuration.



Select Data Source

Select an option that this wizard will use to obtain data about this relying party:

( ) Import data about the relying party published online or on a local network

Use this option to import the necessary data and certificates from a relying party organization that publishes its federation metadata online or on a local network.

Federation metadata address (host name or URL):

Example: fs.contoso.com or https://www.contoso.com/app

( ) Import data about the relying party from a file

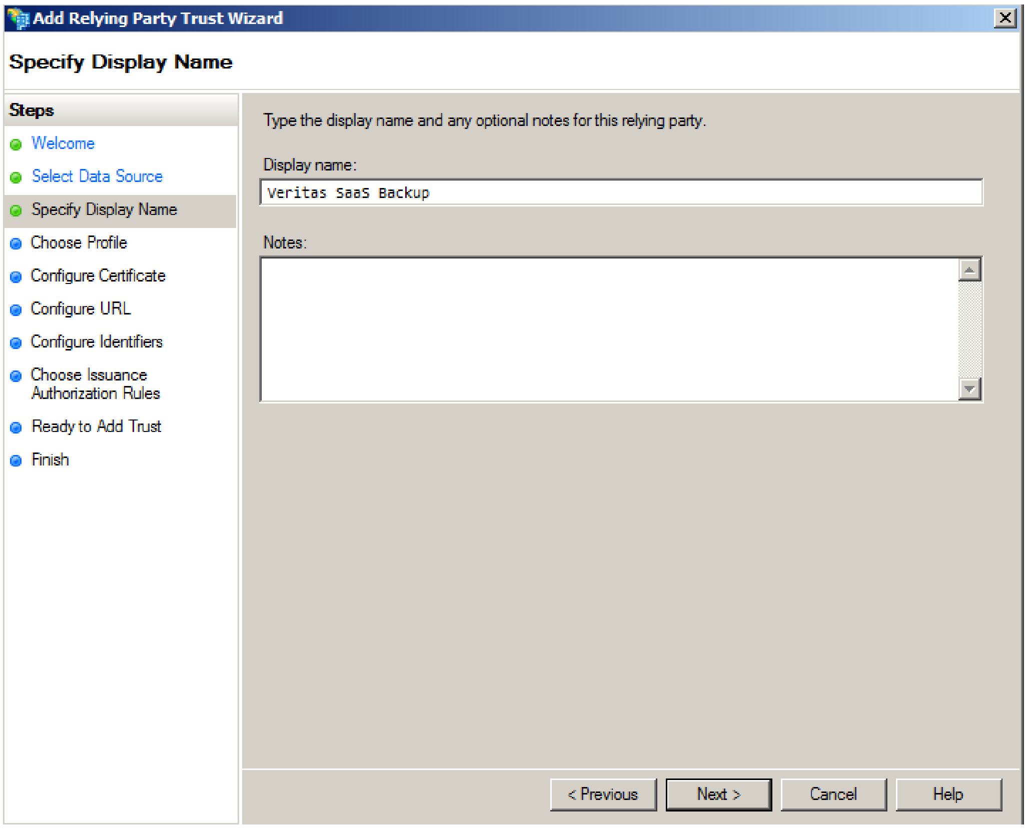
Use this option to import the necessary data and certificates from a relying party organization that has exported its federation metadata to a file. Ensure that this file is from a trusted source. This wizard will not validate the source of the file.

Federation metadata file location:

[✓ Enter data about the relying party manually]

Use this option to manually input the necessary data about this relying party organization.

3. Specify display name of the relying party.



Specify Display Name

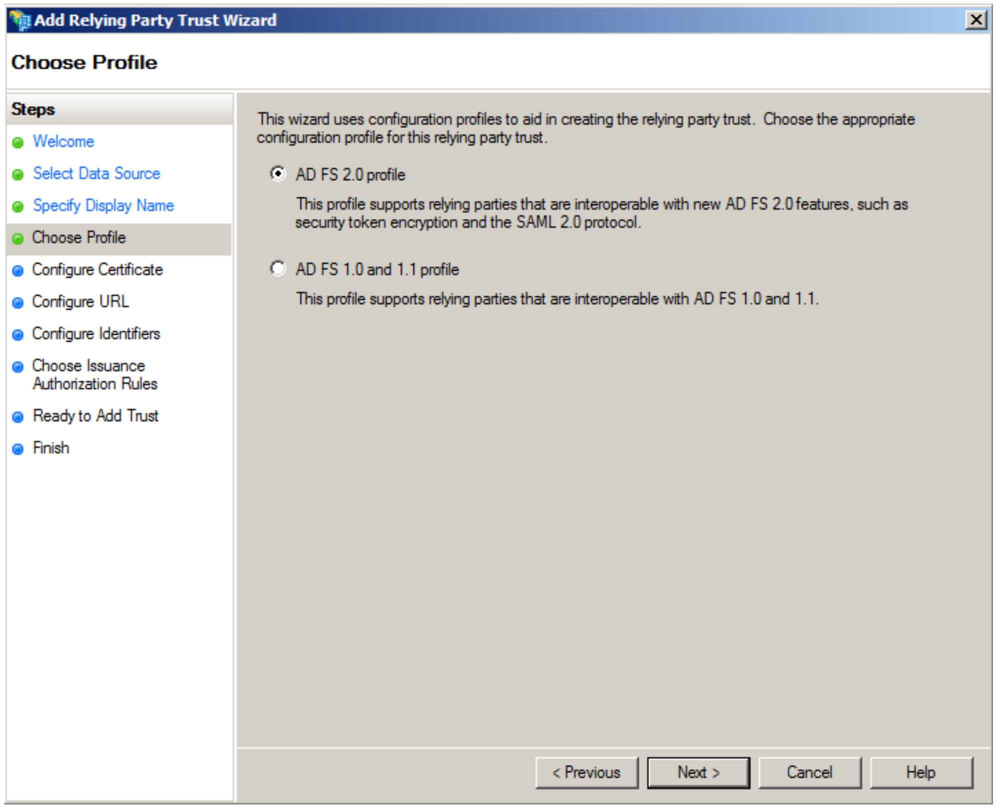
Type the display name and any optional notes for this relying party.

Display name:

```
Veritas SaaS Backup
```

Notes:

4. Choose ADFS 2.0 profile and don't specify certificate.



This wizard uses configuration profiles to aid in creating the relying party trust. Choose the appropriate configuration profile for this relying party trust.

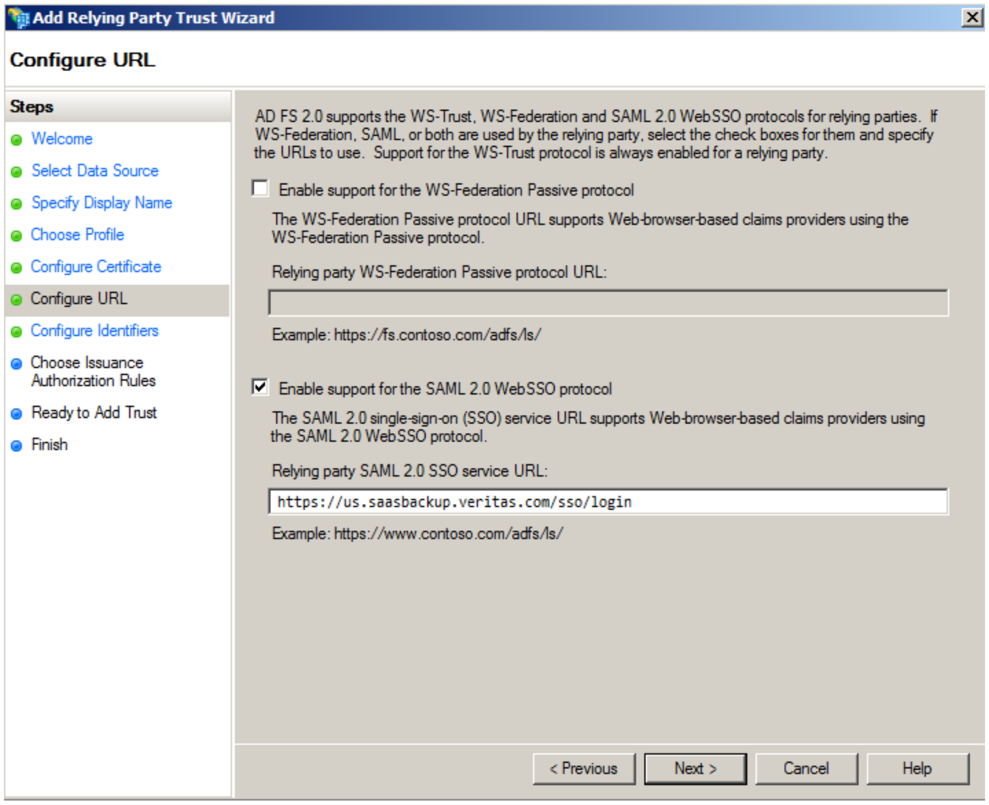
[✓ AD FS 2.0 profile]

This profile supports relying parties that are interoperable with new AD FS 2.0 features, such as security token encryption and the SAML 2.0 protocol.

AD FS 1.0 and 1.1 profile

This profile supports relying parties that are interoperable with AD FS 1.0 and 1.1.

5. In the configure url section, enable support for the SAML 2.0 Web SSO protocol and add the following URL for Relying party SAML 2.0 SSO service URL



AD FS 2.0 supports the WS-Trust, WS-Federation and SAML 2.0 WebSSO protocols for relying parties. If WS-Federation, SAML, or both are used by the relying party, select the check boxes for them and specify the URLs to use. Support for the WS-Trust protocol is always enabled for a relying party.

☐ Enable support for the WS-Federation Passive protocol

The WS-Federation Passive protocol URL supports Web-browser-based claims providers using the WS-Federation Passive protocol.

Relying party WS-Federation Passive protocol URL:

Example: https://fs.contoso.com/adfs/ls/

☑ Enable support for the SAML 2.0 WebSSO protocol

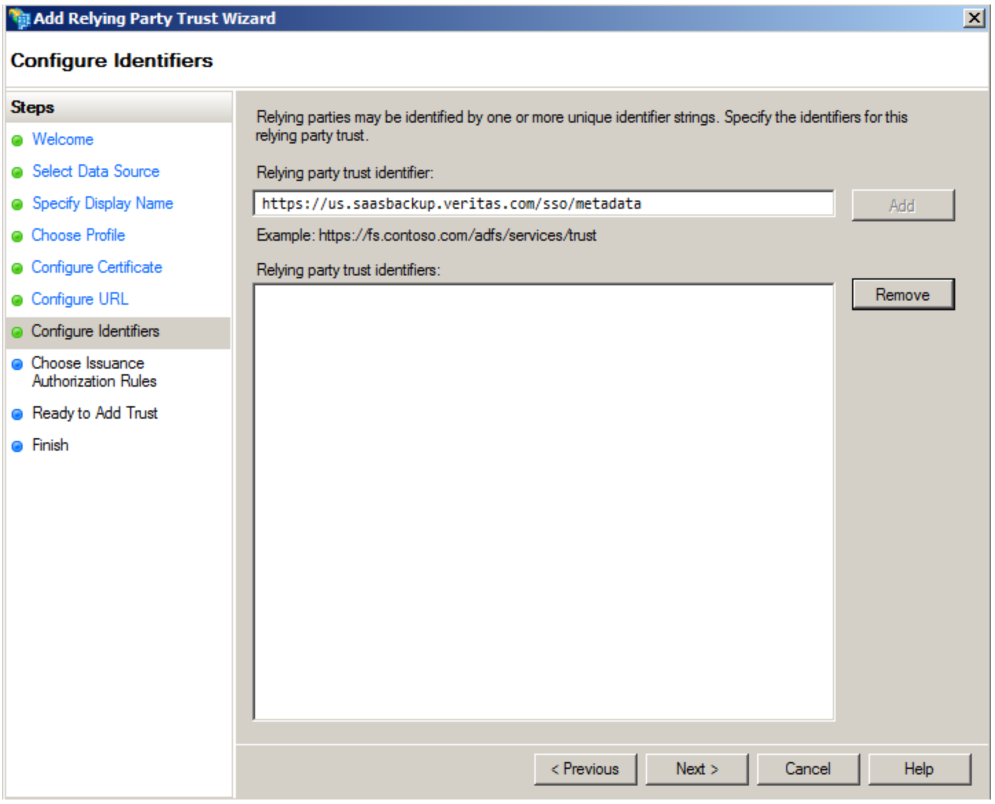
The SAML 2.0 single-sign-on (SSO) service URL supports Web-browser-based claims providers using the SAML 2.0 WebSSO protocol.

Relying party SAML 2.0 SSO service URL:

`https://us.saasbackup.veritas.com/sso/login`

Example: https://www.contoso.com/adfs/ls/

6. Add Veritas as trust identifier.



Add Relying Party Trust Wizard

Configure Identifiers

Steps
- Welcome
- Select Data Source
- Specify Display Name
- Choose Profile
- Configure Certificate
- Configure URL
- Configure Identifiers
- Choose Issuance Authorization Rules
- Ready to Add Trust
- Finish

Relying parties may be identified by one or more unique identifier strings. Specify the identifiers for this relying party trust.

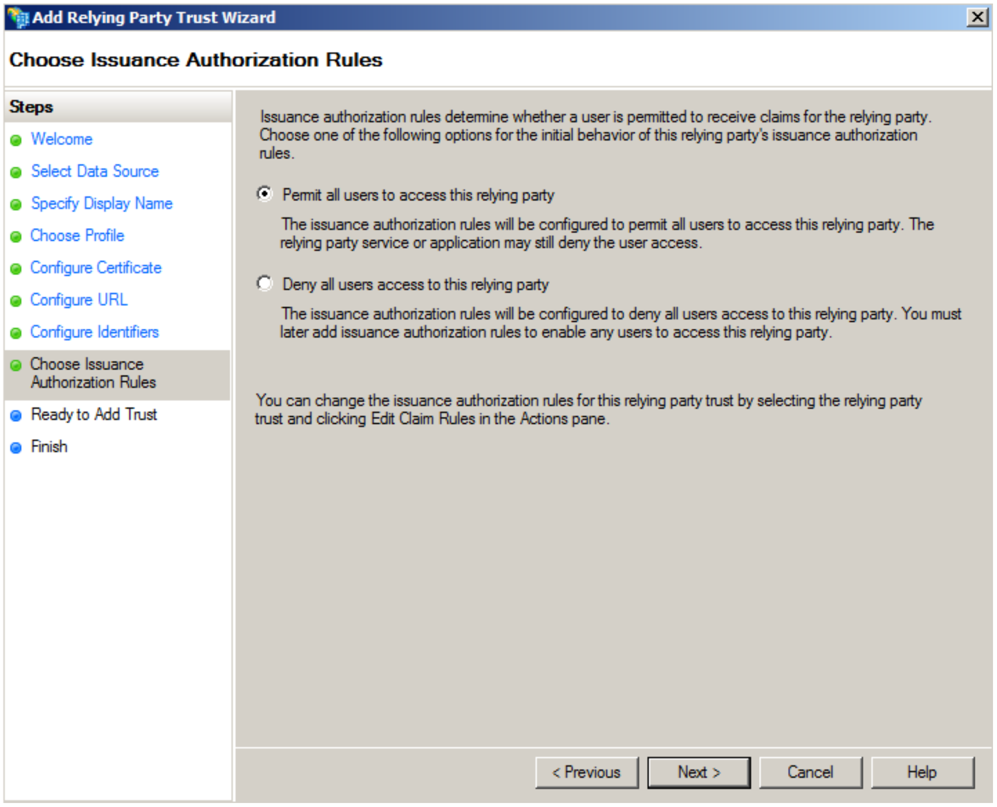
Relying party trust identifier:

`https://us.saasbackup.veritas.com/sso/metadata`

Example: https://fs.contoso.com/adfs/services/trust

Relying party trust identifiers:

7. Permit all users to access Veritas SaaS Backup relying party.



Add Relying Party Trust Wizard

Choose Issuance Authorization Rules

Steps
- Welcome
- Select Data Source
- Specify Display Name
- Choose Profile
- Configure Certificate
- Configure URL
- Configure Identifiers
- Choose Issuance Authorization Rules
- Ready to Add Trust
- Finish

Issuance authorization rules determine whether a user is permitted to receive claims for the relying party. Choose one of the following options for the initial behavior of this relying party's issuance authorization rules.

[✓ Permit all users to access this relying party]

The issuance authorization rules will be configured to permit all users to access this relying party. The relying party service or application may still deny the user access.

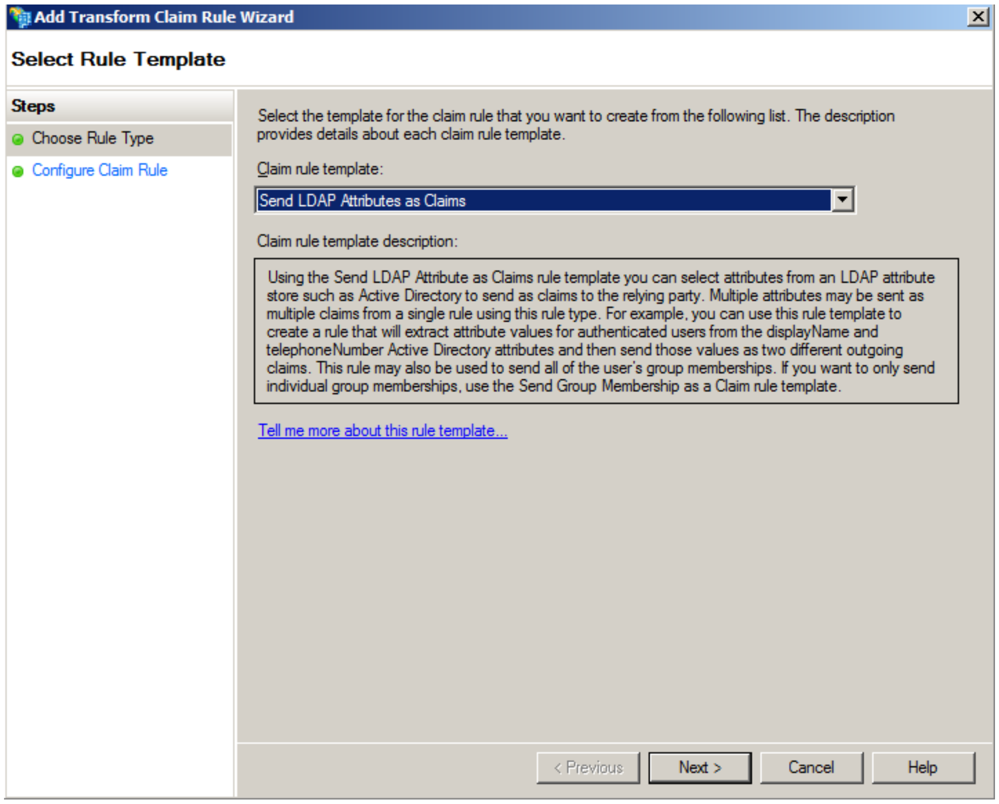
Deny all users access to this relying party

The issuance authorization rules will be configured to deny all users access to this relying party. You must later add issuance authorization rules to enable any users to access this relying party.

You can change the issuance authorization rules for this relying party trust by selecting the relying party trust and clicking Edit Claim Rules in the Actions pane.

8. At the last step of this wizard check "Open the Edit Claim Rules dialog".

9. In the edit claim rules dialog add rule from template "Send LDAP Attributes as Claims".


Add Transform Claim Rule Wizard
Steps
Choose Rule Type
Configure Claim Rule
Tell me more about this rule template...
< Previous
Next >
Cancel
Help

Select Rule Template

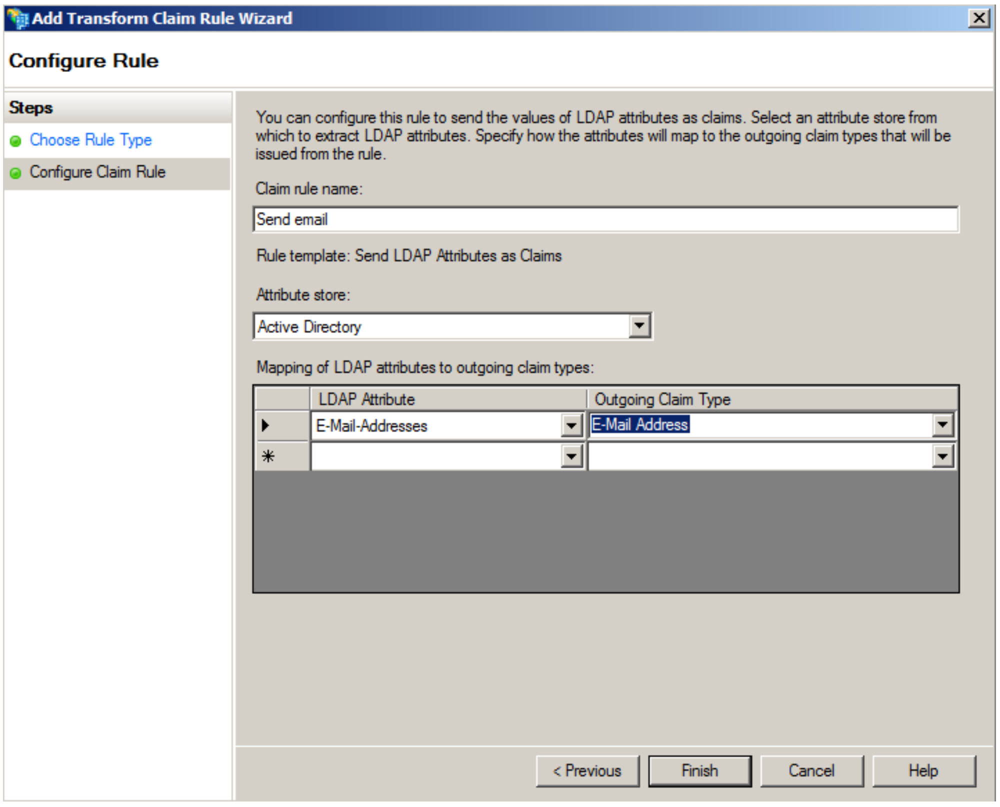
Select the template for the claim rule that you want to create from the following list. The description provides details about each claim rule template.

Claim rule template: Send LDAP Attributes as Claims

Claim rule template description:

Using the Send LDAP Attribute as Claims rule template you can select attributes from an LDAP attribute store such as Active Directory to send as claims to the relying party. Multiple attributes may be sent as multiple claims from a single rule using this rule type. For example, you can use this rule template to create a rule that will extract attribute values for authenticated users from the displayName and telephoneNumber Active Directory attributes and then send those values as two different outgoing claims. This rule may also be used to send all of the user's group memberships. If you want to only send individual group memberships, use the Send Group Membership as a Claim rule template.

10. Set name to "Send email", select "Active Directory" as an attribute store and map "E-Mail-Addresses" LDAP attribute to "E-Mail Address" outgoing claim type.



Configure Rule

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name: Send email

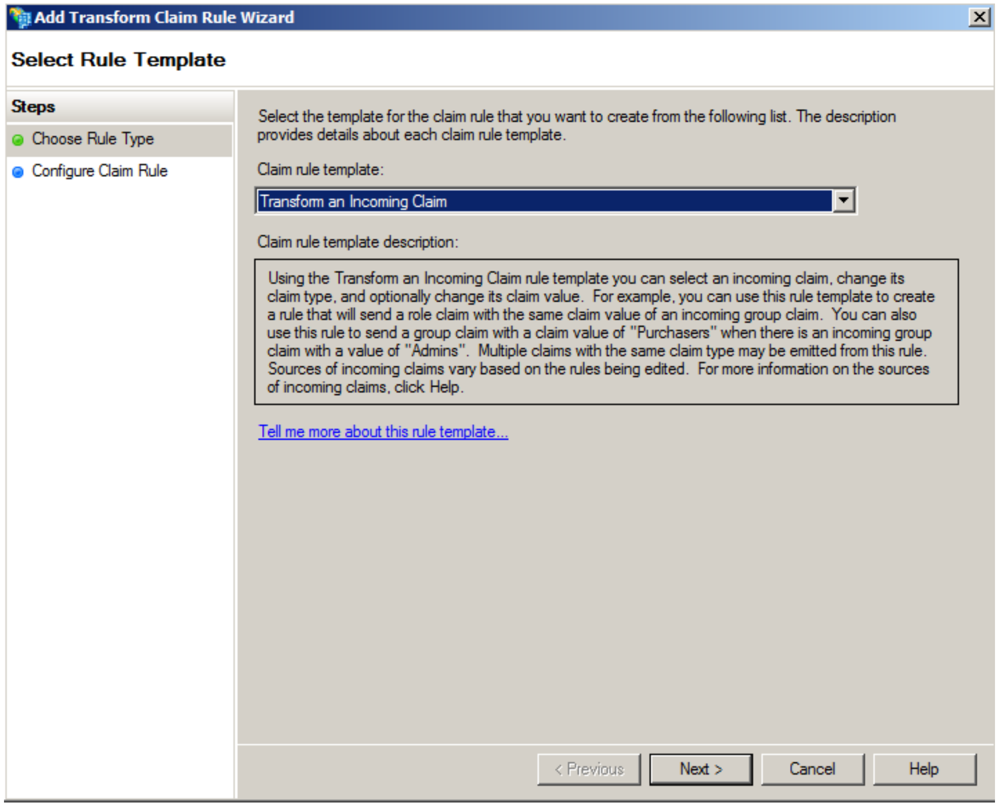
Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

	LDAP Attribute	Outgoing Claim Type
▶	E-Mail-Addresses	E-Mail Address
*		

11. In the edit claim rules dialog add a new rule from template "Transform an Incoming Claim".



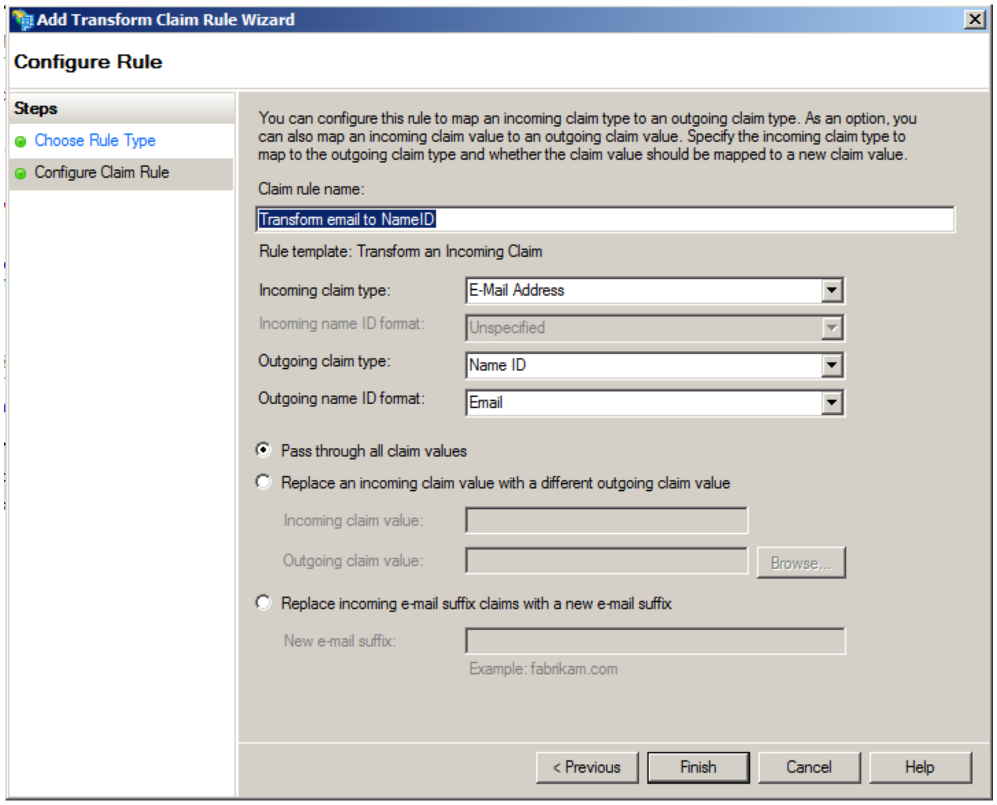
Select the template for the claim rule that you want to create from the following list. The description provides details about each claim rule template.

Claim rule template: Transform an Incoming Claim

Claim rule template description:

Using the Transform an Incoming Claim rule template you can select an incoming claim, change its claim type, and optionally change its claim value. For example, you can use this rule template to create a rule that will send a role claim with the same claim value of an incoming group claim. You can also use this rule to send a group claim with a claim value of "Purchasers" when there is an incoming group claim with a value of "Admins". Multiple claims with the same claim type may be emitted from this rule. Sources of incoming claims vary based on the rules being edited. For more information on the sources of incoming claims, click Help.

12. Set "Transform email to NameID" as rule name, "E-Mail Address" as incoming rule type, "Name ID" as outgoing claim type and "Email" as outgoing name id type.



You can configure this rule to map an incoming claim type to an outgoing claim type. As an option, you can also map an incoming claim value to an outgoing claim value. Specify the incoming claim type to map to the outgoing claim type and whether the claim value should be mapped to a new claim value.

Claim rule name: Transform email to NameID

Rule template: Transform an Incoming Claim

Incoming claim type: E-Mail Address

Incoming name ID format: Unspecified

Outgoing claim type: Name ID

Outgoing name ID format: Email

(●) Pass through all claim values

( ) Replace an incoming claim value with a different outgoing claim value

Incoming claim value:

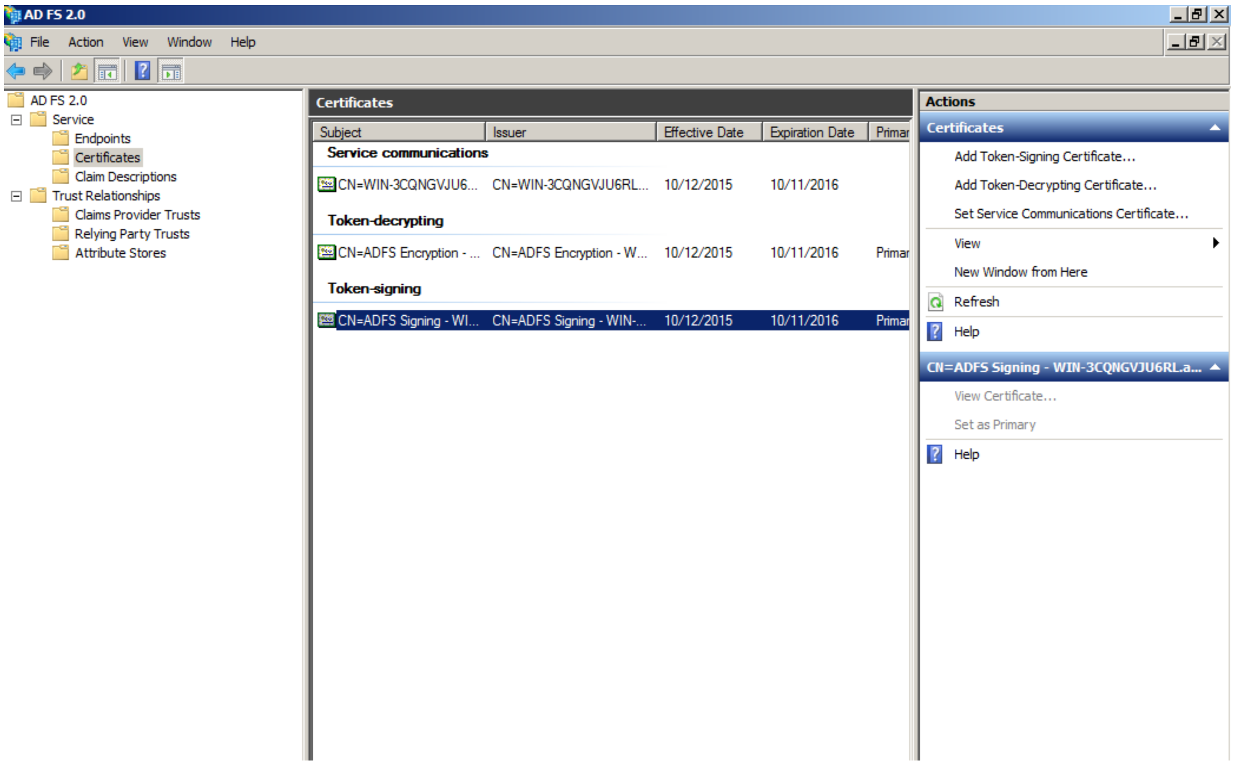
Outgoing claim value:

( ) Replace incoming e-mail suffix claims with a new e-mail suffix

New e-mail suffix:

Example: fabrikam.com

13. Save created rules.
14. Save AD FS token signing certificate into file in .pem format

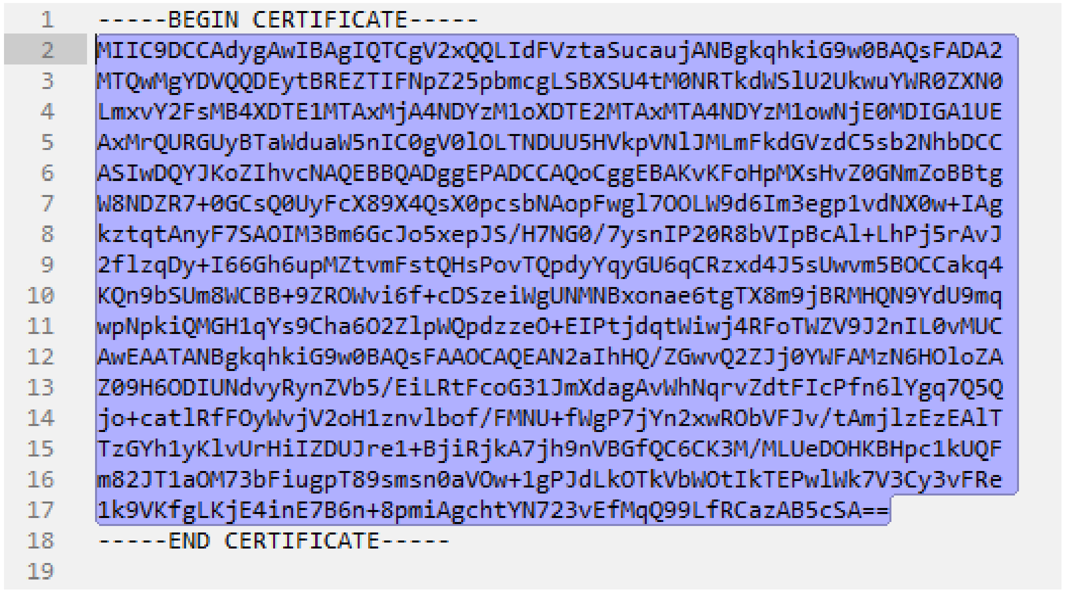


Certificates

Subject	Issuer	Effective Date	Expiration Date	Primar
Service communications				
CN=WIN-3CQNGVJU6...	CN=WIN-3CQNGVJU6RL...	10/12/2015	10/11/2016	
Token-decrypting				
CN=ADFS Encryption - ...	CN=ADFS Encryption - W...	10/12/2015	10/11/2016	Primar
Token-signing				
CN=ADFS Signing - WI...	CN=ADFS Signing - WIN-...	10/12/2015	10/11/2016	Primar

15. Use the content of this file as a certificate of your identity provider in Veritas SaaS Backup SSO configuration.

    Note: Copy certificate excluding certificate begin/end markers



```
-----BEGIN CERTIFICATE-----
MIIC9DCCAdygAwIBAgIQTCgV2xQQLIdFVztaSucaujANBgkqhkiG9w0BAQsFADA2
MTQwMgYDVQQDEytBREZTIFNpZ25pbmcgLSBXSU4tM0NRTkdWSlU2UkwuYWR0ZXN0
LmxvY2FsMB4XDTE1MTAxMjA4NDYzM1oXDTE2MTAxMTA4NDYzM1owNjE0MDIGA1UE
AxMrQURGUyBTaWduaW5nIC0gV0lOLTNDUU5HVkpVNlJMLmFkdGVzdC5sb2NhbDCC
ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKvKFoHpMXsHvZ0GNmZoBBtg
W8NDZR7+0GCsQ0UyFcX89X4QsX0pcsbNAopFwgl7OOLW9d6Im3egp1vdNX0w+IAg
kztqtAnyF7SAOIM3Bm6GcJo5xepJS/H7NG0/7ysnIP20R8bVIpBcAl+LhPj5rAvJ
2flzqDy+I66Gh6upMZtvmFstQHsPovTQpdyYqyGU6qCRzxd4J5sUwvm5BOCCakq4
KQn9bSUm8WCBB+9ZROWvi6f+cDSzeiWgUNMNBxonae6tgTX8m9jBRMHQN9YdU9mq
wpNpkiQMGH1qYs9Cha6O2ZlpWQpdzzeO+EIPtjdqtWiwj4RFoTWZV9J2nIL0vMUC
AwEAATANBgkqhkiG9w0BAQsFAAOCAQEAN2aIhHQ/ZGwvQ2ZJj0YWFAMzN6HOloZA
Z09H6ODIUNdvyRynZVb5/EiLRtFcoG31JmXdagAvWhNqrvZdtFIcPfn6lYgq7Q5Q
jo+catlRfFOyWvjV2oH1znvlbof/FMNU+fWgP7jYn2xwRObVFJv/tAmjlzEzEAlT
TzGYh1yKlvUrHiIZDUJre1+BjiRjkA7jh9nVBGfQC6CK3M/MLUeDOHKBHpc1kUQF
m82JT1aOM73bFiugpT89smsn0aVOw+1gPJdLkOTkVbWOtIkTEPwlWk7V3Cy3vFRe
1k9VKfgLKjE4inE7B6n+8pmiAgchtYN723vEfMqQ99LfRCazAB5cSA==
-----END CERTIFICATE-----
```

Configuration of Veritas SaaS Backup

SSO can be configured by Veritas administrator accounts. Administrator can create any number of SSO configurations, but the first enabled configuration from the list will be used by the Veritas SaaS Backup servers.

All subaccounts of this administrator account will be able to use SSO once it's configured.

The next parameters must be provided:

- SSO configuration name - distinguished name of the configuration, unique for given account.
- Prefix - will be used for customer subdomains i.e. customer.sso.saasbackup.veritas.com. Must be unique in the whole system.
- Identity provider URL - SSO sign in URL of the identity provider.
- Certificate - X.509 certificate of the identity provider in .pem format.

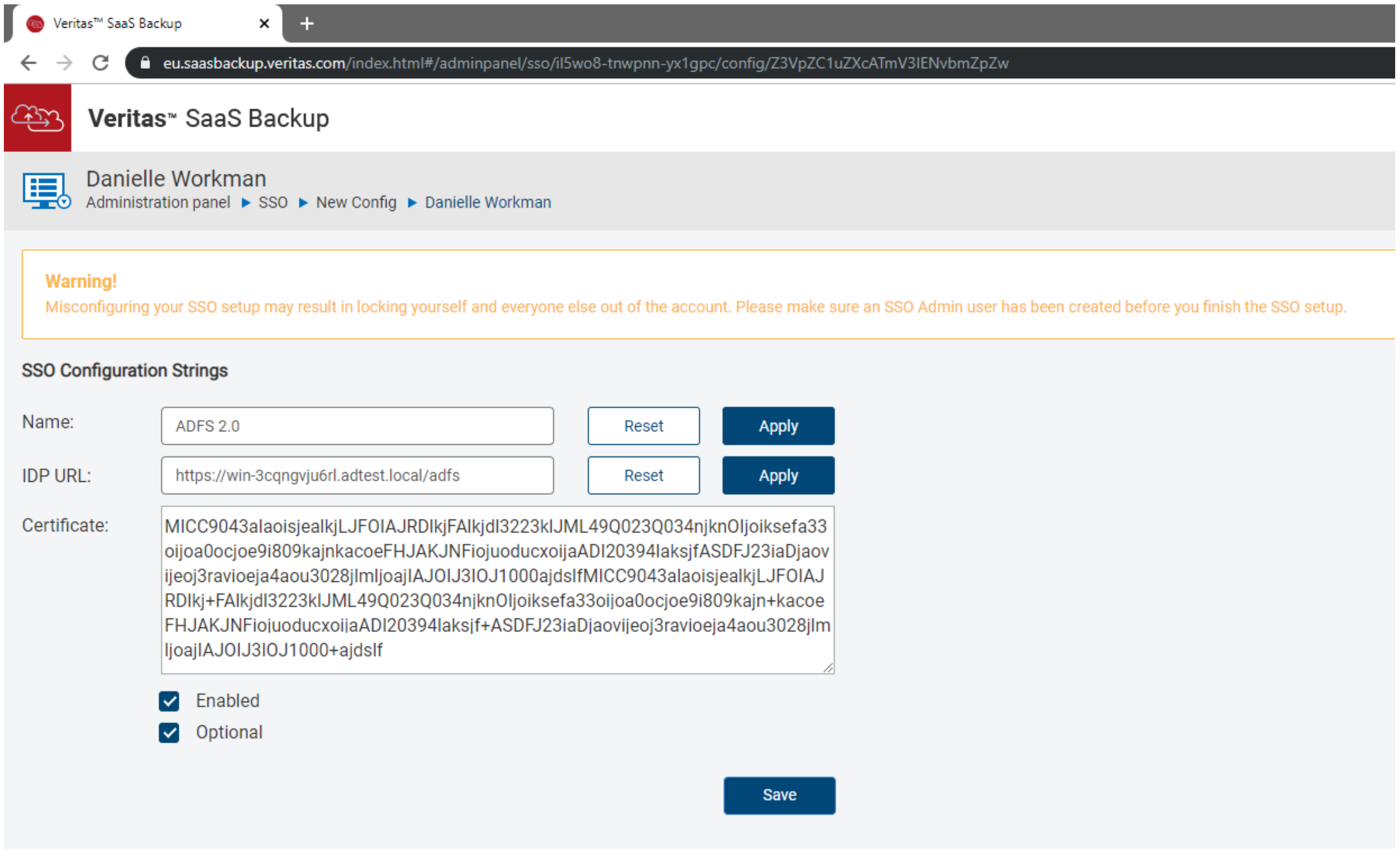
Here is some documentation we have on VSB setup with SSO:

- How to configure Single Sign-on(SSO) with Veritas SaaS Backup - https://www.veritas.com/support/en_US/article.100045126
- How to create an SSO admin account - https://www.veritas.com/content/support/en_US/article.100045464.html

Sample configuration

This is the Veritas part of AD FS 2.0 sample setup.

Note: Sign in url of AD FS 2.0 server is https://<adfs-server-name>/adfs/ls


Veritas™ SaaS Backup
eu.saasbackup.veritas.com/index.html#/adminpanel/sso/il5wo8-tnwpnn-yx1gpc/config/Z3VpZC1uZXcATmV3IENvbmZpZw
Veritas™ SaaS Backup
Danielle Workman
Administration panel ▶ SSO ▶ New Config ▶ Danielle Workman
Reset
Apply
Save

Warning!
Misconfiguring your SSO setup may result in locking yourself and everyone else out of the account. Please make sure an SSO Admin user has been created before you finish the SSO setup.

SSO Configuration Strings

Name: ADFS 2.0

IDP URL: https://win-3cqngvju6rl.adtest.local/adfs

Certificate:

```
MICC9043alaoisjealkjLJFOIAJRDIkjFAlkjdl3223klJML49Q023Q034njknOljoiksefa33
oijoa0ocjoe9i809kajnkacoeFHJAKJNFiojuoducxoijaADI20394laksjfASDFJ23iaDjaov
ijeoj3ravioeja4aou3028jlmljoajlAJOIJ3IOJ1000ajdslfMICC9043alaoisjealkjLJFOIAJ
RDIkj+FAlkjdl3223klJML49Q023Q034njknOljoiksefa33oijoa0ocjoe9i809kajn+kacoe
FHJAKJNFiojuoducxoijaADI20394laksjf+ASDFJ23iaDjaovijeoj3ravioeja4aou3028jlm
ljoajlAJOIJ3IOJ1000+ajdslf
```

[✓] Enabled
[✓] Optional

API Description

Method GET /users/{user-id}/ssoconfig

This method retrieves all user's SSO configurations. Only one configuration can be active at a time. If user has several enabled configurations, only the first will be used.

Response

The response body will adhere to the following schema

```
element configurations {
  element configuration {
    element guid { text }
    & element name { text }
    & element prefix { text } # host prefix for this configuration i.e. test results in
test.sso.us.saasbackup.veritas.com
    & element idp_url { text }  # address of the identity provider endpoint
    & element certificate { text } # .pem idp certificate
    & element enabled { boolean }
    & element optional { boolean } # login with Veritas SaaS Backup password is allowed
if set to true
  }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	OK	FAIL	FAIL	FAIL	FAIL

Method POST /users/{user-id}/ssoconfig

This method creates a new SSO configurations. Only one configuration can be active at a time. If user has several enabled configurations, only the first will be used.

Request

The request body must adhere to the following schema

```
element configuration {
  element name { text }
  & element prefix { text } # host prefix for this configuration i.e. test results in
test.sso.us.saasbackup.veritas.com
  & element idp_url { text }  # address of the identity provider endpoint
  & element certificate { text } # .pem idp certificate
  & element enabled { boolean }
  & element only_subaccounts { boolean } # Apply this configuration only to subaccounts
but not current account itself
  & element optional { boolean }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	OK	FAIL	FAIL	FAIL	FAIL

Method GET /users/{user-id}/ssoconfig/{ssoconfig-id}

This method retrieves given SSO configuration.

Response

The response body will adhere to the following schema

```
element configuration {
  element name { text }
  & element prefix { text } # host prefix for this configuration i.e. test results in
test.sso.us.saasbackup.veritas.com
  & element idp_url { text }  # address of the identity provider endpoint
  & element certificate { text } # .pem idp certificate
  & element enabled { boolean }
  & element only_subaccounts { boolean } # Apply this configuration only to subaccounts
but not current account itself
  & element optional { boolean }
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	OK	FAIL	FAIL	FAIL	FAIL

Method PUT /users/{user-id}/ssoconfig/{ssoconfig-id}

This method updates SSO configuration.

Request

The request body must adhere to the following schema:

```
element configuration {
  element name { text } ?
  & element prefix { text } ? # host prefix for this configuration i.e. test results in
test.sso.us.saasbackup.veritas.com
  & element idp_url { text } ? # address of the identity provider endpoint
  & element certificate { text } ? # .pem idp certificate
  & element enabled { boolean } ?
  & element only_subaccounts { boolean }? # Apply this configuration only to
subaccounts but not current account itself
  & element optional { boolean } ?
}
```

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	OK	FAIL	FAIL	FAIL	FAIL

Method DELETE /users/{user-id}/ssoconfig/{ssoconfig-id}

This method deletes SSO configuration.

ACL

Public	Partner	Admin	User	Device	System	NONE
FAIL	FAIL	OK	FAIL	FAIL	FAIL	FAIL

SSO (not REST) endpoints

Method GET /sso/metadata

This method retrieves metadata of Veritas SaaS Backup SP.

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	OK

Method GET /sso/login?uid={user-email}&code={code}

This method returns URL of user's IdP containing all necessary information to run authentication request. URL is returned in plain text. Caller must check return code. 200 is returned on success and 404 if SSO configuration couldn't be resolved. SSO configuration is resolved by using GET request parameters.

For example, `GET https://us.saasbackup.veritas.com/sso/login?uid=user@veritas.com&code=1` uses IdP from the SSO configuration set for user with primary token user@saasbackup.veritas.com.

Parameter 'code' is an integer flag. If it's set to 0 user is finally redirected to /index.html and to /ssoticket.html otherwise.

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	OK

Method POST /sso/login

This method is called automatically by the IdP and processes its SAML response(XML document). It:

- Verifies document's signature
- Looks for user id which must be placed in NameID tag in urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress format

- Matches user id with Veritas SaaS Backup account
- Creates one-time shortliving token (SSO ticket) which is returned in cookies and used to obtain standard Veritas tokens later

POST body contains 2 important parameters:

- SAMLResponse where the response from identity provider is stored
- RelayState this parameter is sent by the proxy to IdP and eventually returned back. It contains information about used SSO configuration and the location where user will be finally redirected. RelayState is url and base64-encoded string of format "sso-configuration-guid:code", where code is an integer flag. If it's set to 0 user is finally redirected to /index.html and to /ssoticket.html otherwise.

This method redirects user either to /index.html or /ssoticket.html and passes one-time short-living SSO ticket in cookie named "ssouserid". This cookie contains user id (usually email) and SSO access code (ticket) in format "userid:sso-ticket". This information is either displayed on /ssoticket.html page or used as user login and password for obtaining regular Veritas tokens.

ACL

Public	Partner	Admin	User	Device	System	NONE
OK	OK	OK	OK	OK	OK	OK

Volatile event queueing API

Purpose of volatile queues

To implement synchronization, clients need to notify other clients that a new backup has been performed on some specific drive.

If the clients are all listening to messages from some queue, any client posting a message to the queue can notify the others of such simple changes. The queue can be volatile, in that it does not need to record events or be able to replay them later. Any client that gets disconnected will simply re-subscribe and re-initialize its state.

Theory of operation

Any client can subscribe to a named queue.

Any client can post events to a named queue.

There is no way to inquire of the existence of a queue.

Events posted will consist of an event type (a short readable UTF8 type name) and some event data (plain UTF8 text string).

The event type "reset" is a reserved event type which is used internally in client implementations to signal a loss of connectivity with the queue service - therefore events with event type "reset" cannot be posted.

Subscribing to a queue

We use Server-Sent Events messaging.

The client will issue this request:

```
GET /vqueues/{queue-id}[?events=ev0,ev1,...]
```

The server will respond with this head:

```
200 OK
transfer-encoding: chunked
content-type: text/event-stream
```

Whenever someone posts an event to the given queue-id, matching the optional event types filter, the server will send a chunk holding

```
event: ...given event type...
data: ...given data...
```

If we cannot write to a client - eg. if writing to the client blocks for too long, we simply close the connection. The client will then know to re-connect and re-initialize.

Posting to a queue

Anyone can post to a queue by issuing a

```
POST /vqueues/{queue-id}/{event-type}
```

The request body must be a plain UTF8 text string with the event data.

What the server does when it receives a post, is, that it re-issues this requests to its sibling API servers, and then it posts the event to any of its locally connected clients who happen to subscribe to the given queue (optionally matching the event filter).

Queue existence and lifetime

The queues do not exist as actual message queues in memory or otherwise. Messages are transmitted (they may be queued in byte stream buffers in the application or in the OS but are not otherwise queued) to clients directly. If a client cannot keep up it gets disconnected.

What this means, is that as long as the API servers re-posts received events to all their API server siblings, we do not need to care about which clients subscribe to which queues on which API servers. Everything will get delivered eventually (or it will fail due to API server restart or connection close - but it will fail in a way the client will notice).

Queuing API

Method GET /vqueues/{queue-id}

Subscribes to a given queue

Method POST /vqueues/{queue-id}/{event}

Posts an event of the given type to the given queue.

Request body must contain plain UTF8 data - the event data.

Queues in current use

The following queues are currently in use in the system:

Queue name	Signalled by	Purpose
history-{device-guid}	proxy	Tells web UI, sync clients and others that the /history for the given device has been changed (usually this means a new backup was added)
bsearch-{device-guid}	buslog	Tells web UI and other UIs that the bsearch results for the given device may have changed (this queue is signalled whenever one or more new snapshots have been imported)
devices-{account-guid}	proxy	Tells web UI, sync clients and others that the device list for a given user account has changed.
history-*	proxy	Requires System token authentication for subscription! An event is posted on this queue whenever any of the history-{device-guid} queues get an event. It is used for the business logic for accounting etc. Each "update" event will have a data line containing "account-guid device-guid device-type".
devices-*	proxy	Requires System token for sub. Event is posted whenever a device-{account-guid} queue gets an event. Used by BL. Each update event will have a data line containing "account-guid".
queue-post-{event}	proxy	When welcome_mail or forgotpass events are put in the /queue/, we also signal the VMQ. This allows business logic to effectively have a subscription on a reliable queue.
user-resource-*	buslog	Whenever the effectively assigned resources are changed for some user account (due to product re-assignment, product configuration change or date change causing a new configuration to take effect), the potentially affected user guids are sent with "update" events on this queue
search-{account-guid}	search	When new content has been indexed for a particular account-guid an "update" event will be trigger, with the payload being the device-guid.
attributes-*	proxy	Informs that a global attribute has been changed
attributes-a-b-c	proxy	Informs that the specified global attribute 'a/b/c' has been changed

ABOUT VERITAS TECHNOLOGIES LLC

Veritas World Headquarters

2625 Augustine Drive

Santa Clara, CA 95054

+1 (866) 837 4827

www.veritas.com

For specific country offices
and contact numbers,
please visit our website.

VERITAS™
The truth in information.