ORACLE

Oracle Eloqua

Developer Guide

30-May-2024

Contents

Oracle Eloqua

Developer Guide

Contents

Getting started with Oracle Eloqua APIs

This section contains information to help you get started developing for the Eloqua platform.

Authenticate using HTTP basic authentication

> ❑ **Warning**: For security reasons we recommend authentication using OAuth 2.0. Learn about OAuth 2.0

For HTTP basic authentication, each request must include an authentication header, with a base-64 encoded value.

Your authentication token is of the format:

```
siteName + '\' + username + ':' + password
```

Where `siteName` is the company name you use to log in to Eloqua, and `username` and `password` are your Eloqua username and password.

For example, for a user whose company name is COMPANYX, username is user1, and password is password123, the base string is:

```
COMPANYX\user1:password123
```

Running base-64 encoding on that string gives the token for the Authentication header: `Q09NUEFOWVhcdXNlcjE6cGFzc3dvcmQxMjM=`

The authentication header would then be:

```
Authorization: Basic Q09NUEFOWVhcdXNlcjE6cGFzc3dvcmQxMjM=
```

Authenticate using OAuth 2.0

OAuth 2.0 enables the safe retrieval of secure resources while protecting user credentials. Some apps may need to authenticate during the configuration phase and others may need OAuth only when a user invokes a service.

In general, OAuth authentication follows a six step pattern:

1. An application requests authorization on a user's behalf.
2. The application obtains a *Grant Token*.
3. The client requests an access token by using the *Grant Token*.
4. The authorization server validates the *Grant Token* and issues an *Access Token* and a *Refresh Token*.
5. The client requests the protected resource, authenticating using the *Access Token*.
6. The resource server verifies the *Access Token* and serves the request.

Learn more about the OAuth 2.0 Specifications.

In this topic:

Authorization endpoint: `https://login.eloqua.com/auth/oauth2/authorize`

Token endpoint: `https://login.eloqua.com/auth/oauth2/token`

Note: The following examples assume that all requests are successful. For a detailed description of possible request responses, including a variety of failure types, see: the OAuth Reference.

To authenticate using OAuth 2.0

Eloqua supports three possible flows that an application can use to obtain access on behalf of a resource owner: **Authorization Code** grant, **Implicit** grant, **Resource Owner Password Credentials** grant. In general, you should use the *Authorization Code* grant for Apps that extend Eloqua's functionality.

Important: Before you begin, you need a unique *Client ID* and *Client Secret* for your app. In your Eloqua instance navigate to **Settings** > **AppCloud Developer** then select **Create New App**. Fill out the required information and enter your app's URL into the *Callback Url* field under the *OAuth* heading then click **Save**. You should then be presented with valid *Client Id* and *Client Secret* values.

To authenticate using an authorization code grant:

1. Request initial authorization through the `login.eloqua.com/auth/oauth2/authorize` endpoint. A call to this endpoint will trigger a prompt for users to enter their credentials.

`/auth/oauth2/authorize` has five possible URL parameters:

Parameter	Value	Required?
`response_type`	Must be "`code`"	Yes
`client_id`	Your app's *Client Id* provided when registering your app (see above)	Yes
`redirect_uri`	Your app's registered redirection endpoint, should be the same URL you entered as the *Callback Url* when registering your app (see above)	Yes
`scope`	Must be "`full`" or not supplied	No
`state`	An optional value that has meaning for your App	No

The call to the `authorize` endpoint might resemble:

```
https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=a1b2c3d4&redirect_uri=https://client.example.com/cb
&scope=full&state=xyz
```

Once users enter their credentials and accept your app's request to access Eloqua on their behalf, they are redirected to the `redirect_uri` with a *Grant Token* (which is in this case an *Authorization Code*) attached in the `code` URL parameter, as in the following example:

```
HTTP/1.1 302 Found
Location:
https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz
```

2. Use the *Grant Token* to obtain an *Access Token* and *Refresh Token* using a `POST` request to the `login.eloqua.com/auth/oauth2/token` endpoint.

The `POST` request should include a JSON body with the following parameters:

Parameter	Value	Required?
`grant_type`	The name of the *Grant Token*'s type. In this case: `authorization_code`	Yes
`code`	The *Grant Token*	Yes
`redirect_uri`	Your app's registered redirection endpoint	Yes

The following example call requests an *Access Token* and a *Refresh Token* token using the *Grant Token* obtained previously:

```
POST https://login.eloqua.com/auth/oauth2/token
Authorization: Basic Q09NUEFOWVhcdXNlcjE6cGFzc3dvcmQxMjM=
    {
        "grant_type":"authorization_code",
        "code":"SplxlOBeZQQYbYS6WxSbIA",
        "redirect_uri":"https://client.example.com/cb"
    }
```

Note: This request must authenticate using HTTP basic. Use your app's *Client Id* as the username and its *Client Secret* as the password. The format is `client_id:client_secret`. Encode the string with base-64 encoding, and you can pass it as an authentication header. The system does not support passing *Client Id* and *Client Secret* parameters in the JSON body, and, unlike basic authentication elsewhere, you should not include your site name. Learn more about basic authentication with Eloqua.

The authorization server validates the authorization code and if valid responds with a JSON body containing the *Access Token*, *Refresh Token*, access token expiration time, and token type, as in the following example:

```
HTTP/1.1 200 OK
Content-Type: application/json {
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"bearer",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

3. Store and use the access and refresh tokens.

   When your app needs a protected resource, it authenticates during the request using the *Access Token*. The following call to Eloqua's application API uses the access token to authenticate:

   ```
   GET /resource/1 HTTP/1.1
   Host: api.eloqua.com
   Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
   ```

   `api.eloqua.com` verifies the *Access Token*, and supplies the requested resource if the access token is valid.

Note:

- *Authorization Codes* expire in **60 seconds** (intended for immediate use)
- *Access Tokens* expire in **8 hours**
- *Refresh Tokens* expire in **1 year**
- *Refresh Tokens* will expire **immediately** after being used to obtain new tokens, or after **1 year** if they are not used to obtain new tokens

4. If the access token has expired, you should send your *Refresh Token* to `login.eloqua.com/auth/oauth2/token` to obtain new tokens as in the following example:

```
POST https://login.eloqua.com/auth/oauth2/token
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnlxS3REUmJuZlZkbUl3
{
    "grant_type":"refresh_token",
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
    "scope":"full",
    "redirect_uri":"https://client.example.com/cb"
}
```

If the request is successful, the response is a JSON body containing a new access token, token type, access token expiration time, and new refresh token:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"bearer",
```

```
    "expires_in":3600,
    "refresh_token":"MToxLUIyZHRNTUZsazIwNmZFTy1"
}
```

Store the new refresh token as the old refresh token is no longer valid. Then proceed to make your call using the new access token.

To authenticate using an implicit grant:

An implicit grant is a browser-based authentication best used for in-browser applications and instead of issuing an authorization code that is exchanged for an access token, Eloqua issues an access token directly.

❑ **Warning**: An authorization code grant is recommended in the vast majority of cases as it has fewer potentially negative security implications.

1. Request an access token through a `GET` request to the `login.eloqua.com/auth/oauth2/authorize` endpoint using the following URL parameters:

Parameter	Value	Required?
`response_type`	Must be "`token`"	Yes
`client_id`	Your app's *Client Id* provided when registering your app (see above)	Yes
`redirect_uri`	Your app's registered redirection endpoint, should be the same URL you entered as the *Callback Url* when registering your app (see above)	Yes

Parameter	Value	Required?
`scope`	Must be "`full`" or not supplied	No
`state`	An optional value that has meaning for your App	No

The call to the `authorize` endpoint might resemble:

```
https://login.eloqua.com/auth/oauth2/authorize?response_type=token
&client_id=s6BhdRkqt3&redirect_uri=https://client.example.com/app
&scope=full&state=xyz
```

Once users enter their credentials and accept your app's request to access Eloqua on their behalf, they are redirected to the `redirect_uri` with an authorization code attached in the `access_token` URL parameter, as in the following example:

```
HTTP/1.1 302
Found Location: https://client.example.com/cb#access_
token=2YotnFZFEjr1zCsicMWpAA
&token_type=bearer&expires_in=28800&state=xyz
```

2. Use the access token to request resources on the user's behalf.

To authenticate using a resource owner password credentials grant

With the resource owner password credential grant, the App provides the resource owner's username and password to the authorization server in order to obtain an access token. This grant type is ideal when an app already has the necessary Eloqua user names and passwords stored, and wants to use access tokens for added security.

1. Obtain an *Access Token* and a *Refresh Token* by making a `POST` request to the `login.eloqua.com/auth/oauth2/token` endpoint using a JSON body containing the

following parameters:

Parameter	Value	Required?
`grant_type`	Must be "`password`"	Yes
`scope`	Must be "`full`" or not supplied	No
`username`	The user's site name and username in the form `sitename` + '`/`' + `username`	Yes
`password`	The user's password	Yes

Note: This request must authenticate using HTTP basic. Use your app's *Client Id* as the username and its *Client Secret* as the password. The format is `client_id:client_secret`. Encode the string with base-64 encoding, and you can pass it as an authentication header. The system does not support passing *Client Id* and *Client Secret* parameters in the JSON body, and, unlike basic authentication elsewhere, you should not include your site name. Learn more about basic authentication with Eloqua.

The call to the `token` endpoint might resemble:

```
POST https://login.eloqua.com/auth/oauth2/token
Authorization: Basic Q09NUEFOWVhcdXNlcjE6cGFzc3dvcmQxMjM=
{
    "grant_type":"password",
    "scope":"full",
    "username":"testsite\\testuser",
    "password":"user123"
```

```
}
```

2. If the request is successful, the response is a JSON body containing an *Access Token*, a *Refresh Token*, the access token expiration time, and token type:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"bearer",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKW"
}
```

3. You can then use the *Access Token* and store the *Refresh Token*, using the *Refresh Token* as needed to renew the credentials. Follow steps 3 and 4 of "To authenticate using an authorization code grant" above for more information.

Troubleshooting error messages

401 Unauthorized

401 Unauthorized errors are divided into categories indicated by the 1000s digit of the `error_code`.

Error code	Description
1000	General error messages.
2000	Error messages related to authentication.
2500	Error messages related to OAuth2 authentication.
3000	Error messages related to authorization.

Here is a list of error messages when submitting requests to the token endpoint `login.eloqua.com/auth/oauth2/token`.

Error	Error code	Error Description
unknown_error	2500	An unknown error occurred.
unknown_token	2501	Provided token is unknown.
unknown_site_id	2502	Provided site identifier is unknown.
destroyed_token	2503	The supplied refresh token has been destroyed.
expired_token	2504	The supplied refresh token has expired.
invalid_client_secret	2505	The supplied client secret doesn't match the client's secret.
unsupported_site_authentication	2506	The site doesn't support OAuth 2 authentication.
unsupported_user_authentication	2507	The user doesn't support OAuth 2 authentication.
unknown_client_id	2508	The supplied client identifier is unknown.
invalid_redirect_uri	2509	The supplied RedirectUri is invalid.
unknown_error	3000	Unknown authorization error.
account_disabled	3001	User account is disabled.
failed_allowlist_authorization	3002	Failed site IP allow list authorization.
unknown_site_id	3003	The supplied site identifier is unknown.
unknown_authentication_handle	3004	The supplied authentication handle identifier is unknown.
unknown_user_id	3005	The supplied user identifier is unknown.
unknown_security_domain	3006	The supplied users security domain is unknown.
invalid_authentication_handle	3007	The supplied authentication handle is invalid.
invalid_request	3008	The request is invalid.

Too Many Requests

The Eloqua OAuth 2.0 authorization flow initiation will only accept one initiation per minute for any given User of an App.

- **For Using Code Grant and Using Implicit Grant**: When user clicks Accept a Too Many Requests message is returned.
- **For Password Credentials Grant**: The following response will be returned to the request:

```
HTTP/1.1 429 Too Many Requests
Content-Type: application/json
Retry-After: 60
{
  "error":"too_many_requests",
  "error_description":"This user has already authorized this app within the allowed time frame of 60 seconds."
}
```

OAuth Responses: Authorization Code Grant Request

Acceptance

If the user accepts your App's request to access Eloqua on their behalf, their user agent is eventually redirected to your app's redirection endpoint with an authorization code in the `code` URL parameter, as in the following example authorization dialog:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz
```

Rejection

If the user rejects your app's request to access Eloqua on their behalf, their user agent is eventually redirected to your App's registered redirection endpoint with the error `access_denied` in the `error` URL parameter, as in the following:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=access_denied&state=xyz
```

Failure Before `client_id` or `redirect_url` Validation

If a failure occurs before the supplied client_id or redirect_uri are validated, we can't safely redirect the user agent back to the redirect URI to report the failure, and so we return the details of the failure in the body of the response.

Missing `client_id`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&redirect_uri=https%3a%2f%2fclient.example.com%2fapp&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;client_id&amp;quot; parameter is required.
```

Unknown `client_id`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=00000000000000000000000000000000
&redirect_uri=https%3a%2f%2fclient.example.com%2fapp&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;client_id&amp;quot; value is not a known client identifier.
```

Malformed `client_id`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=malformed&redirect_uri=https%3a%2f%2fclient.example.com%2fapp
&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;client_id&amp;quot; value is not a valid client identifier.
```

Missing `redirect_uri`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;redirect_uri&amp;quot; parameter is required.
```

Malformed `redirect_uri`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&redirect_uri=malformed&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;redirect_uri&amp;quot; value is not a valid URI.
```

Mismatched `redirect_uri`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&redirect_uri=https%3a%2f%2attacker.com%2fapp
&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;redirect_uri&amp;quot; value doesn&amp;#39;t start with the client redirect URI.
```

Non-HTTPS `redirect_uri`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&redirect_uri=http%3a%2f%2fclient.example.com%2fapp
&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;redirect_uri&amp;quot; value is not an HTTPS URI.
```

`redirect_uri` with fragment

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&redirect_
uri=https%3a%2f%2fclient.example.com%2fapp%23fragment
&scope=full&state=xyz
```

```
HTTP/1.1 200 OK
Content-Type: text/html

The &amp;quot;redirect_uri&amp;quot; value has a fragment.
```

Failure After `client_id` and `redirect_uri` Validation

If a failure occurs after the `client_id` and `redirect_uri` have been validated, Eloqua can safely redirect user agent back to the redirect URI to report the failure. In this case, the Authorization Dialog returns the details of the failure in the `error` and `error_description` URL parameters.

Internal server error

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=server_error
&error_description=The+server+encountered+an+unexpected+condition+that+prevented
```

```
+it+from+fulfilling+the+request.&state=xyz
```

Missing `response_type`

```
GET https://login.eloqua.com/auth/oauth2/authorize?
client_id=s6BhdRkqt3&redirect_uri=https%3a%2f%2fclient.example.com%2fapp
&scope=full&state=xyz

HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=invalid_request
&error_description=The+%22response_type%22+parameter+is+required.&state=xyz
```

Unknown `response_type`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=unknown
&client_id=s6BhdRkqt3&redirect_uri=https%3a%2f%2fclient.example.com%2fapp
&scope=full&state=xyz

HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=unsupported_response_type
&error_description=The+%22response_
type%22+parameter+must+be+either+%22code%22
+or+%22token%22.&state=xyz
```

Unknown `scope`

```
GET https://login.eloqua.com/auth/oauth2/authorize?response_type=code
&client_id=s6BhdRkqt3&redirect_uri=https%3a%2f%2fclient.example.com%2fapp
&scope=unknown&state=xyz

HTTP/1.1 302 Found
Location: https://client.example.com/cb?error=invalid_scope
&error_description=The+%22scope%22+parameter+must+be+either+%22full%22+or
+not+supplied.&state=xyz
```

OAuth signing

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system, in this case your app, can validate that the call was sent by Eloqua. As an app provider, it is your responsibility to ensure the validity of all inbound calls. Validation should be performed on every inbound call your app receives with OAuth parameters.

When your application receives any call from Eloqua, it will be appended by the following information:

Parameter	Description	Example value
`oauth_consumer_key`	Your app's *Client Id*.	eb954432-a19f-4250-85dd-827a9ddf17db
`oauth_nonce`	A random unique number used by the app provider to verify that a request has never been made before, preventing replay attacks. Nonces only need to be unique for all requests using the same time stamp.	9519484
`oauth_signature_method`	Eloqua uses a keyed-hash message authentication specification known as HMAC-SHA1 to sign outgoing calls.	HMAC-SHA1
`oauth_timestamp`	The timestamp is expressed in UTC in UNIX format, expressed as the number of seconds since January	1410986606

Parameter	Description	Example value
	1, 1970 00:00:00 GMT. The timestamp value must be no more than 5 minutes older than your current server time.	
`oauth_version`	1.0	1.0
`oauth_signature`	This is the value against which to validate.	AZbD26DeXrEV6iNLqBAxSXwWURg=

Example: Eloqua makes a call to your app situated at `https://app.example.com/action/create` in order to pass some information: `instance_id=768acf98-f0d2-4f1b-8956-bd204de20684&site_id=b379a93e-dd7a-41a1-99be-fffd93c8e4fa`. This HTTP call would look something like:

```
https://app.example.com/action/create?instance_id=768acf98-f0d2-4f1b-8956-bd204de20684&site_id=b379a93e-dd7a-41a1-99be-fffd93c8e4fa&oauth_consumer_key=eb954432-a19f-4250-85dd-827a9ddf17db&oauth_nonce=9519484&oauth_signature_method=HMACSHA1&oauth_timestamp=1410986606&oauth_version=1.0&oauth_signature=xoEGUaC029gD8UWeEOyguxGBkZU%3D
```

Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

Validating a call signature

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system, in this case your app, can validate that the call was sent by Eloqua. As an app provider, it is your

responsibility to ensure the validity of all inbound calls. Validation should be performed on every inbound call your app receives with OAuth parameters.

When your app receives a call from Eloqua, there are certain steps that must be taken to validate it. To illustrate these steps, let's say a POST request is received by your app from this URL:

```
https://example.com/eloqua/action/create?Special!Character=test@test&AssetName=Campaign+With+Spaces&oauth_consumer_key=test_client_id&oauth_nonce=1234567&oauth_signature_method=HMAC-SHA1&oauth_timestamp=1427308921&oauth_version=1.0&oauth_signature=WeeqclooECjp2LEGPlkabKVhkEo%3D
```

Note: Any spaces in the URL are changed to a "+". For example, the Asset Name in Eloqua of "Campaign With Spaces" will be "Campaign+With+Spaces" in the request.

To validate a call signature:

1. Select the `oauth_consumer_key` value and ensure it matches your app's *Client Id* found in your app's configuration under **Settings** > **AppCloud Developer** in your Eloqua instance. If this value does not match your app's *Client Id* the call is invalid and should be discarded.
2. Select the `oauth_timestamp` value and check that it is no more than 5 minutes older than your current server time. The timestamp will be in UTC in UNIX format (the number of seconds since January 1, 1970 00:00:00 GMT). If this value is more than 5 minutes old the call is invalid and should be discarded.

3. Select the `oauth_nonce` value and compare it to `oauth_nonce` values for calls with the same `oauth_timestamp`. If it matches any other `oauth_nonce` values from a call with the same `oauth_timestamp`, the request is invalid and should be discarded.

4. Cache the `oauth_nonce` and the corresponding `oauth_timestamp` values for 5 minutes so you can check the nonce against future calls.

5. Calculate the first chunk of the signature base string. The signature base string consists of three parts, separated by ampersands ("&"):
    - The HTTP method (GET, POST, PUT, DELETE, etc.)
    - The URL endpoint without query string parameters, percent-encoded
    - The query parameters, omitting the oauth_signature parameter, sorted alphabetically, percent-encoded

    Using the `https://example.com...` example (see above), the first chunk of the signature base string, the HTTP method, is: `POST`.

6. Calculate the second chunk of the signature base string. The second chunk consists of the URL endpoint without query parameters.

> **Important**: You need to ensure the URL does not include the port number. For example: `https://example.com` and not `https://example.com:443`. Notably, Java EE's native `HttpServletRequest.getRequestURL()` function returns a URL including the protocol, server name, port number, and server path.

Using the above example we get: `https://example.com/eloqua/action/create`. This URL endpoint then needs to be percent-encoded. Using the above example we get:

```
https%3A%2F%2Fexample.com%2Feloqua%2Faction%2Fcreate
```

> **Important**: You need to ensure this string is encoded with percent codes and *uppercase* letters. For example, `https://example.com` should encode to `https%3A%2F%2Fexample.com` and not `https%3a%2f%2fexample.com`. Case sensitivity is not important in regular HTTP transport, but it is important when hashing to generate a signature. Notably, .NET's native `HttpUtility.UrlEncode()` function encodes strings with lower-case percent codes -- these will need to adjusted manually by looping through the string and ensuring the two characters after every percentage sign ("%") are in uppercase.

7. Calculate the third chunk of the signature base string. The third chunk consists of the call's query parameters. Make sure not to include the question mark between endpoint URL and query string and to remove the oauth_signature parameter. Using the above example we get:

```
Special!Character=test@test&AssetName=Campaign+With+Spaces&oauth_
consumer_key=test_client_id&oauth_nonce=1234567&oauth_signature_
method=HMAC-SHA1&oauth_timestamp=1427308921&oauth_version=1.0
```

   Then sort the parameters alphabetically. Using the above example we get:

```
AssetName=Campaign+With+Spaces&oauth_consumer_key=test_client_id&oauth_
nonce=1234567&oauth_signature_method=HMAC-SHA1&oauth_
timestamp=1427308921&oauth_version=1.0&Special!Character=test@test
```

   Before percent-encoding the whole URL change any "+" to "%20", and percent-encode any special characters in all parameter keys and values. Using the above example you'd get:

```
AssetName=Campaign%20With%20Spaces&oauth_consumer_key=test_client_
id&oauth_nonce=1234567&oauth_signature_method=HMAC-SHA1&oauth_
timestamp=1427308921&oauth_version=1.0&Special%21Character=test%40test
```

Finally, percent-encode the above URL to get the third chunk of the signature base string. Using the above example we get:

```
AssetName%3DCampaign%2520With%2520Spaces%26oauth_consumer_
key%3Dtest_client_id%26oauth_nonce%3D1234567%26oauth_signature_
method%3DHMAC-SHA1%26oauth_timestamp%3D1427308921%26oauth_
version%3D1.0%26Special%2521Character%3Dtest%2540test
```

8. Complete the signature base string by using ampersands to concatenate the three signature base string chunks: HTTP method, URL endpoint and query parameters. Using the above example we get the final signature base sting:

```
POST&https%3A%2F%2Fexample.com%2Feloqua%2Faction%2Fcreate&AssetName
%3DCampaign%2520With%2520Spaces%26oauth_consumer_key%3Dtest_client_
id%26oauth_nonce%3D1234567%26oauth_signature_method%3DHMAC-
SHA1%26oauth_timestamp%3D1427308921%26oauth_
version%3D1.0%26Special%2521Character%3Dtest%2540test
```

> **Note**: There should be two and only two ampersands in the resultant string. Any ampersands in the query string should have been percent-encoded into **%26** strings,

and any equal signs in the query string should have been percent-encoded into **%3D** strings.

This signature base string will be used as your hash *Message* in subsequent validation steps. The following steps require a SHA1 cryptographic hash function (HMAC-SHA1) to create a keyed-hash message authentication code (known as the *Message*). Most programming languages offer easy-to-use libraries which accept a *Message* and *Key* to create the SHA1 hash.

Important: With Eloqua, the hash *Message* = the call's signature base string and the *Key* = your app's *Client Secret* with an appended ampersand.

Note: Any special characters that were in parameter keys or values, and any "+" that were changed to a "%20", will be double encoded. For example, the "+" that was changed to a "%20" will now be "%2520".

9. Create a hash *Key* by appending an ampersand to your app's *Client Secret*. Your app's *Client Secret* can be found in your app's configuration under **Settings** > **AppCloud Developer** in your Eloqua instance. In your real-world app the *Client Secret* will be a string consisting of 100 random characters. In our example however, we'll say the *Client Secret* is: `test_client_secret`. For the purposes of creating the authentication code, you must append an ampersand to your app's Client Secret, so in this example we get: `test_client_secret&`

10. Enter your appropriately formatted *Message* and *Key* values into a SHA1 hash compute function and run. The computed HMAC value is commonly returned as a Hex value. In the above example, the following Hex value is returned:

    ```
    59e7aa708a281028e9d8b1063e591a6ca561904a
    ```

11. Convert the computed HMAC Hex value to Base64 string. Converting the Hex value for our example (59e7aa708a281028e9d8b1063e591a6ca561904a) results in:

    ```
    WeeqclooECjp2LEGPlkabKVhkEo=
    ```

12. Decode the percent-encoded oauth_signature value (WeeqclooECjp2LEGPlkabKVhkEo%3D). For our example, the decoded oauth_signature value is:

    ```
    WeeqclooECjp2LEGPlkabKVhkEo=
    ```

13. Compare to computed HMAC value that's been converted to Base64 string to the decoded oauth_signature value. If the values match, the call is valid and your app can use it freely. If the values do not match the request is invalid and should be discarded.

Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

Sending API requests

A basic application and Bulk API request has several components not including authentication. An Oracle Eloqua application and Bulk API request follows the form:

`[`**`Verb`**`] [baseURL]/API/[APIName]/[APINumber]/[endpoint]`

Where...

- `baseURL` is the location of your data center. See Determining base URLs for more information.
- `[APIName]` is the name of the API, such as "rest" or "bulk"
- `[APINumber]` is the version of the API, such as "1.0" or 2.0"

Therefore an example of a complete request would be as follows:

```
GET https//secure.<pod>.eloqua.com/API/rest/1.0/assets/emails
```

The reporting API requests follow the form:

```
GET [base URL]/API/odata/[subjectArea]/
[APIVersionIdentifier]/[endpoint]
```

Where `[subjectArea]` is the subject area of the endpoint, such as "campaignAnalysis", and `[APIVersionIdentifier]` is the version of the API, such as "1.0".

Here is an example of a complete reporting API request:

```
GET
https://secure.p03.eloqua.com/api/odata/campaignAnalysis/1.0/
campaign
```

Learn more about the reporting API.

Learn more by watching the video

Determining Base URLs

Before you can send a request, you must determine the Eloqua data center for which your instance is located. Your data center is a component of your API request, if continuing our example above, the data center is the `<pod>` component of your request. Eloqua supports multiple data centers, and the

`https://login.eloqua.com/id` endpoint allows you to interface with Eloqua regardless of where the Eloqua install is located. It is important to validate your base URL before making any API calls. If you don't validate, your API calls may not work.

Learn about base URLs

HTTP verbs

Eloqua's API services support four different methods or "verbs": **POST**, **GET**, **PUT** and **DELETE**. In terms of database functions, these HTTP verbs correspond to the basic operations of "CRUD": create, read, update and delete.

- **GET** is used to retrieve a representation or API entity from a resource. Can be used to retrieve one or more entities. The reporting API only supports GET requests.
- **POST** is used to create a new resource.
- **PUT** is used to update a resource.
- **DELETE** is used to delete a resource.

Learn about HTTP verbs

HTTP request headers

Multiple request headers may be required for API requests to Eloqua's APIs, depending on the endpoint. The documentation for that endpoint will explain which headers are required. Generally, the request headers for most Eloqua endpoints are similar.

Learn about request headers

Request depth

Eloqua application and Bulk APIs can retrieve entities at three different levels of depth: minimal, partial, and complete. In general, requests at minimal depth will perform best, as they scan the least amount of data. For most application API endpoint properties, if a property in the root of the response does not have a value, that property will not be returned.

Learn about request depth

> **Note**: The reporting API does not support request depth. Learn more about reporting API query options.

URL parameters

Application and Bulk request endpoints may also contain one or more URL parameters. Parameters are used to customize requests so as to more easily manipulate information in the database.

Learn more: URL parameters

HTTP status codes

In most cases Eloqua's APIs will respond to requests with the standard HTTP status code definitions. If the API fails to validate a request, it will respond with a validation error message (JSON or XML) describing the issue.

Learn about HTTP status codes

Bulk API status codes

In addition to the standard HTTP status codes, Eloqua includes specific status codes to help inform you when performing operations with the their APIs. Each status code has a number, as well as a human-readable message.

Learn about bulk API status codes

Validation errors

If the API fails to validate a request, it will respond with a 400 validation error message (JSON or XML) describing the issue.

Learn about validation errors

Endpoints

Endpoints allow access to Oracle Eloqua resources and operations. Each resource/operation is associated with a specific URL. Endpoints have can be accessed and manipulated through HTTP verbs, but not all verbs are valid for all endpoints.

> **Note**: An endpoint is sometimes referred to as a "URI" or uniform resource identifier.

For example, performing a GET request on the REST 1.0 "`assets/emails`" endpoint will return a list of all the emails in your instance's database. However, you cannot perform a POST request on this endpoint.

See the API reference documentation

For more information about the endpoints beyond the reference material, such as tutorials and walkthroughs, see the links below.

HTTP verbs

There are 5 basic request patterns which apply to most Oracle Eloqua endpoints, and these are based on the HTTP verbs. The basic request patterns are:

- Retrieve a single item (GET)
- Retrieve a list of items (GET)
- Create an item (POST)
- Update an item (PUT)
- Delete an item (DELETE)

Below are examples of each pattern.

> **Note**: The following examples apply to the application and Bulk APIs. For examples of using the reporting API, see Getting started with the reporting API.

Retrieve a single item (GET)

To retrieve a single entity, perform a GET request for that entity's endpoint, optionally specifying the format to return the entity in. The below example retrieves the information for the contact with id#123.

Request:

```
GET https://.../data/contact/123

Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 2164
Content-Type: application/json
X-Request-Id: 34b3d527-d9b6-414d-8967-d9f976bf416d
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Wed,
 10 Jun 2015 19:36:18 GMT

{
   "type":"Contact",
   "currentStatus":"Awaiting action",
   "id":"123",
   "createdAt":"1424362751",
   "depth":"complete",
   "name":"john.a.macdonald@canada.com",
   "updatedAt":"1424362751",
   "accountName":"Government of Canada",
   "address1":"123 Front St.",
   "businessPhone":"011-555-5555",
   "city":"Kingston",
   "emailAddress":"john.a.macdonald@canada.com",
   "emailFormatPreference":"unspecified",
   "fieldValues":[...],
   "firstName":"John",
   "isBounceback":"false",
   "isSubscribed":"true",
   "lastName":"Macdonald",
   "salesPerson":"Chuckles",
   "subscriptionDate":"1424362751",
   "title":"Prime Minister"
```

```
}
```

Retrieve a list of items (GET)

To retrieve a list of entities, perform a GET request for that entity type's list endpoint, specifying the query parameters to use to filter the list, and optionally specifying the format to return the entities in. The below example retrieves a list of all contacts.

> **Note**: Eloqua APIs generally employ different endpoint paths for retrieval of list versus single items. For example: `GET https://.../data/contact/123` retrieves the contact with id#123 while `GET https://.../data/contacts`retrieves a list of all contacts. Notice the second example uses the plural "contacts" rather than the singular "contact" of the first example.

Request:

```
GET https://.../data/contacts
Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 119989
Content-Type: application/json
X-Request-Id: a7dcbbd4-c9b1-43b3-893a-e4094dd2a99b
```

```
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Tue,
 23 Jun 2015 19:17:21 GMT

{
   "elements":[
      {
         "type":"Contact",
         "id":"1",
         "createdAt":"1403034086",
         "depth":"minimal",
         "name":"adeline.wong@oracle.com",
         "updatedAt":"1410193024",
         "emailAddress":"adeline.wong@oracle.com"
      },
      {
         "type":"Contact",
         "id":"2",
         "createdAt":"1403113589",
         "depth":"minimal",
         "name":"demo-/contacts/imports/1-00@oracle.com",
         "updatedAt":"1403113589",
         "emailAddress":"demo-/contacts/imports/1-00@oracle.com"
      },

  ...

  ],
    "page":1,
    "pageSize":1000,
    "total":543
}
```

Create an item (POST)

To create an entity, perform a POST request to that entity's endpoint specifying the entity's send format (Content-Type), and optionally specifying its return format

(Accept). The below example creates a contact with email address and last name values.

Request header:

```
POST https://.../data/contact/
 Content-Type: application/json
Accept: application/json
```

Request body:

```
{
 " emailAddress":"fortinbras@norway.com",
 "lastName": "Fortinbras"
}
```

Response:

```
HTTP/1.1 201 Created
Cache-Control: private
Content-Length: 1995
Content-Type: application/json
X-Request-Id: 9bbee8c7-3522-49ab-8f93-236ce9042910
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Tue,
 23 Jun 2015 19:23:32 GMT

{
   "type":"Contact",
   "currentStatus":"Awaiting action",
   "id":"23648",
```

```
   "createdAt":"1435087412",
   "depth":"complete",
   "name":"fortinbras@norway.com",
   "updatedAt":"1435087412",
   "emailAddress":"fortinbras@norway.com",
   "emailFormatPreference":"unspecified",
   "fieldValues":[...],
   "isBounceback":"false",
   "isSubscribed":"true",
   "lastName":"Fortinbras",
   "subscriptionDate":"1434039531"
}
```

Update an item (PUT)

To update an entity, perform a PUT request to that entity's endpoint, specifying the entity's send format (Content-Type), and optionally specifying its return format (Accept). The below example updates the email address and phone number of the contact with #Id 22482.

Note: the id is a required parameter for PUT requests in both the request header and body (and the two must match). In addition to the id parameter, for contacts, the "emailAddress" is required for the request body, and for many other assets the "name" parameter is required. Please see the documentation for the specific asset you wish to work with for more information on endpoint constraints.

Request header:

```
PUT https://.../data/contact/22482
 Content-Type: application/json
Accept: application/json
```

Request body:

```
{
 "emailAddress":"fortinbras@norway.com",
 "id":"22482",
 "businessPhone":"555-555-5555"
}
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 2052
Content-Type: application/json
X-Request-Id: 347249bc-4fbe-461b-a268-5488d7bc5bae
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

X-Powered-By: ASP.NET
Date: Thu,
 11 Jun 2015 19:39:39 GMT

{
   "type":"Contact",
   "currentStatus":"Awaiting action",
   "id":"22482",
   "initialId":"22482",
   "createdAt":"1434039531",
   "depth":"complete",
```

```
  "name":"fortinbras100@norway.com",
  "updatedAt":"1434051580",
  "businessPhone":"555-555-5555",
  "emailAddress":"fortinbras100@norway.com",
  "emailFormatPreference":"unspecified",
  "fieldValues":[...],
  "isBounceback":"false",
  "isSubscribed":"true",
  "subscriptionDate":"1434050717"
}
```

Delete an item (DELETE)

To delete an entity, perform a DELETE request to that entity's endpoint.

Request:

```
DELETE https://.../data/contact/123
```

HTTP request headers

The sections below outline the request headers used by Oracle Eloqua APIs.

Content-Type

Content-Type specifies the media type that the client is sending to the server. You should always use `application/json`. If no value is supplied, an error will occur. For example:

```
PUT https://.../data/contact/123
Content-Type: application/json
<the existing contact>
```

Important: For PUT and POST verbs, use of the Content-Type header is mandatory.

Accept

Accept specifies the media types that the client is willing to accept from the server. If no value is supplied, or if the supplied value doesn't contain `application/json`, the response will be returned in JSON.

```
GET https://.../data/contact/123 Accept: application/json
```

```
GET https://.../data/contact/123
```

```
GET https://.../data/contact/123 Accept: text/html
```

Note: use of the Accept header is optional.

X-HTTP-Method-Override

Allows HTTP clients that don't support the PUT or DELETE verbs to use GET or POST to update and delete entities. If a verb value is supplied in the header, that value will be used in place of the actual request verb. The following examples are functionally identical:

```
DELETE https://.../data/contact/123
```

```
GET https://.../data/contact/123 X-HTTP-Method-Override:
DELETE
```

Note: use of the X-HTTP-Method-Override header is optional.

X-HTTP-Status-Code-Override

Allows HTTP clients that don't support or expose HTTP status codes, other than HTTP 200 OK, to indicate that the server should always respond with a specific HTTP status code. If a status code value is supplied in the header, that status code will be used in place of the actual request status code. In the following example, the request will always return HTTP 200 OK, regardless of whether or not the actual status is OK:

```
GET https://.../data/contact/123 X-HTTP-Status-Code-Override:
200
```

If a value is supplied for the header, the actual status code will be returned in the X-HTTP-Original-Status-Code response header.

Note: use of theX-HTTP-Status-Code-Override header is optional.

Request depth

Eloqua application and Bulk APIs can retrieve entities at three different levels of depth: minimal, partial, and complete. In general, requests at minimal depth will perform best, as they scan the least amount of data. For most application API endpoint properties, if a property in the root of the response does not have a value, that property will not be returned.

The descriptions below should help you choose the right level of depth for your request.

> **Note**: The reporting API does not support request depth. Learn more about reporting API query options.

Minimal

Only a small number of the entity's properties are returned. Most properties common to most records: the entity name, type and id, and the dates the entity was created, last updated and last accessed.

Minimal depth typically results in the best performance, as the time needed to retrieve the information in minimized.

The following example shows a contact retrieved at minimal depth:

Request:

```
GET https://.../data/contact/1?depth=minimal

Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 169
Content-Type: application/json
X-Request-Id: 88f7d681-17b7-4607-9176-326e14227abc
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Tue,
 23 Jun 2015 19:57:02 GMT

{
   "type":"Contact",
   "id":"1",
   "createdAt":"1403034086",
   "depth":"minimal",
   "name":"erlich.bachman@aviato.com",
   "updatedAt":"1410193024",
   "emailAddress":"erlich.bachman@aviato.com"
}
```

Partial

All of the entity's properties are returned and if the entity is related to other objects, those entities are returned at minimal depth.

The following example shows a contact retrieved at partial depth:

Request:

```
GET https://.../data/contact/1?depth=partial

Accept: application/json
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 358
Content-Type: application/json
X-Request-Id: b0417f68-07c0-4461-8339-42b40647e4d5
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Tue,
 23 Jun 2015 19:58:46 GMT

{
   "type":"Contact",
   "currentStatus":"Awaiting action",
   "id":"1",
   "createdAt":"1403034086",
   "depth":"partial",
   "name":"erlich.bachman@aviato.com",
   "updatedAt":"1410193024",
   "emailAddress":"erlich.bachman@aviato.com",
   "emailFormatPreference":"unspecified",
   "firstName":"erlich",
   "isBounceback":"false",
   "isSubscribed":"true",
   "lastName":"bachman",
   "subscriptionDate":"1403034086"
}
```

Complete

All of the entity's properties are returned and all related entities are returned at complete depth.

The following example shows a contact retrieved at complete depth:

Request:

```
GET https://.../data/contact/1?depth=partial

Accept: application/json
```

Response:

This is the response header and body:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 2059
Content-Type: application/json
X-Request-Id: 94f4b31a-dc0f-4bb9-8756-3421378a085a
P3P: CP="IDC DSP COR DEVa TAIa OUR BUS PHY ONL UNI COM NAV CNT
STA",

Date: Tue,
 23 Jun 2015 19:59:38 GMT

{
   "type":"Contact",
   "currentStatus":"Awaiting action",
   "id":"1",
   "createdAt":"1403034086",
   "depth":"complete",
   "name":"erlich.bachman@aviato.com",
   "updatedAt":"1410193024",
   "emailAddress":"erlich.bachman@aviato.com",
   "emailFormatPreference":"unspecified",
   "fieldValues":[
      {
         "type":"FieldValue",
         "id":"100005"
      },
```

```
{
  "type":"FieldValue",
  "id":"100017"
},
{
  "type":"FieldValue",
  "id":"100023"
},
{
  "type":"FieldValue",
  "id":"100024"
},
{
  "type":"FieldValue",
  "id":"100032",
  "value":"CPLTO000000000001"
},
{
  "type":"FieldValue",
  "id":"100033"
},
{
  "type":"FieldValue",
  "id":"100034"
},
{
  "type":"FieldValue",
  "id":"100035"
},
{
  "type":"FieldValue",
  "id":"100036"
},
{
  "type":"FieldValue",
  "id":"100041"
},
{
  "type":"FieldValue",
  "id":"100043"
```

```
    },
    {
      "type":"FieldValue",
      "id":"100044"
    },
    {
      "type":"FieldValue",
      "id":"100045"
    },
    {
      "type":"FieldValue",
      "id":"100046"
    },
    {
      "type":"FieldValue",
      "id":"100047"
    },
    {
      "type":"FieldValue",
      "id":"100048"
    },
    {
      "type":"FieldValue",
      "id":"100049"
    },
    {
      "type":"FieldValue",
      "id":"100051"
    },
    {
      "type":"FieldValue",
      "id":"100065"
    },
    {
      "type":"FieldValue",
      "id":"100066"
    },
    {
      "type":"FieldValue",
      "id":"100068"
```

```
    },
    {
      "type":"FieldValue",
      "id":"100069"
    },
    {
      "type":"FieldValue",
      "id":"100072"
    },
    {
      "type":"FieldValue",
      "id":"100081"
    },
    {
      "type":"FieldValue",
      "id":"100171",
      "value":"avliato.com"
    },
    {
      "type":"FieldValue",
      "id":"100172",
      "value":"erlich bachman"
    },
    {
      "type":"FieldValue",
      "id":"100174"
    },
    {
      "type":"FieldValue",
      "id":"100175"
    },
    {
      "type":"FieldValue",
      "id":"100176"
    },
    {
      "type":"FieldValue",
      "id":"100177"
    },
    {
```

```
      "type":"FieldValue",
      "id":"100178"
    },
    {
      "type":"FieldValue",
      "id":"100179"
    },
    {
      "type":"FieldValue",
      "id":"100180"
    },
    {
      "type":"FieldValue",
      "id":"100184"
    },
    {
      "type":"FieldValue",
      "id":"100187"
    },
    {
      "type":"FieldValue",
      "id":"100188"
    },
    {
      "type":"FieldValue",
      "id":"100189",
      "value":"erlichbachman"
    },
    {
      "type":"FieldValue",
      "id":"100190"
    },
    {
      "type":"FieldValue",
      "id":"100191"
    },
    {
      "type":"FieldValue",
      "id":"100192"
    },
```

```
      {
         "type":"FieldValue",
         "id":"100193"
      },
      {
         "type":"FieldValue",
         "id":"100194"
      },
      {
         "type":"FieldValue",
         "id":"100195"
      },
      {
         "type":"FieldValue",
         "id":"100197"
      }
   ],
   "firstName":"erlich",
   "isBounceback":"false",
   "isSubscribed":"true",
   "lastName":"bachman",
   "subscriptionDate":"1403034086"
}
```


URL parameters

Oracle Eloqua's application and Bulk API request endpoints may also contain one or more URL parameters. Parameters are used to customize requests so as to more easily manipulate information in the database. Below you will find a list of standard URL request parameters.

Note: To learn about the reporting API query options, see Getting started with the reporting API.

URL parameters are placed after an endpoint in the call's URL. The URL parameter set should be separated from the endpoint by a "?" symbol. If you add more than one URL parameter they should be separated from each other by a "&" symbol. For example:

GET [base URL]`/API/Rest/2.0?page=5&count=30`

Important: Most application and Bulk API endpoints accept one or more of the below standard parameters, but not all endpoints support all parameters.

Name	Description	Constraints
`depth`	Level of detail returned by the request. Learn more about the depth parameter.	**Possible values:** "minimal" "partial" "complete" **Example:** `?depth=complete`
`count`	Maximum number of entities to return	Any whole number between 1 and 1000 inclusive.

Name	Description	Constraints
		Example: `?count=100`
`page`	Specifies which page of entities to return (the count parameter defines the number of entities per page). If the page parameter is not supplied, 1 will be used by default.	Any positive whole number. **Example:** `?page=3&count=10`
`search`	The `search` parameter specifies the search criteria to use to filter the results. The syntax for the search parameter is: `search={term}{operator}{value}`. Learn more about the search URL parameter.	`{term}` is the name of a field or property to filter on, `{operator}` is the comparison operator, and `{value}` is the value to compare the term with. If `{term}` and `{operator}` are not supplied, the term is compared to the value using the equality operator. Searches can be for exact full matches, or partial matches. A "*****" symbol can be used after the `{value}` for partial matches. If there are spaces in the `value`, the `value` needs to be placed in single quotes. Otherwise, single quotes are not required. You can search with fields even if they are not returned at the depth being used.

Name	Description	Constraints
		The following operators are supported on most endpoints: • **=** (Equal To) • **!=** (Not equal to) • **>** (Greater than) • **<** (Less than) • **>=** (Greater than or Equal to) • **<=** (Less than or Equal to) **Example**: `GET .../data/contacts?search=id=1`
`sort`	Specifies the name of the property used to sort the returned entities.	The value depends on the type of entity being sorted, with each entity having its own list of sortable properties. **Example**: `GET .../data/contacts?sort=firstName`
`dir`	Specifies the direction in which to sort the returned entities.	"asc" for ascending or "desc" for descending. **Example**: `GET .../data/contacts?sort=firstName&dir=asc`
`orderBy`	Specifies the field by which list results are ordered, and the direction. The direction will defaullt to ASC if not specified.	Any valid asset parameter field. **Example**: `?orderBy=createdAt`

Name	Description	Constraints
		`?orderBy=createdAt DESC` `?orderBy=createdAt ASC`
`lastUpdatedAt`	When the asset was last updated. Returns deleted assets.**Note**: For the majority of use cases, it is recommended to use `updatedAt` with the `search` URL parameter. For example: `?search='updatedAt>518862600'`	A valid date/time value. **Example**: `?lastUpdatedAt=518862600`
`viewId`	Specify a view id to filter results by view, when retrieving Contact or Account data.	Must be a valid view id, and must be for the same entity, e.g. if retrieving Contact data the view id must correspond to a Contact View. **Example**: `.../data/contact/1?viewId=100002`

Special cases:

- `statusCodeOverride`

  Allows HTTP clients that don't support or expose HTTP status codes other than HTTP 200 OK to indicate that the server should always respond with a specific HTTP status code. If a value is supplied for the parameter, that value will be used as the status code.

  An example can been seen in the request below, which will always return HTTP 200 OK, regardless of whether or not the actual status is OK:

  ```
  GET https://.../data/contact/123?statusCodeOverride=200
  ```

Eloqua status codes

In addition to the standard HTTP status codes, Eloqua includes specific status codes to help inform you when performing operations with the their APIs. Each status code has a number, as well as a human-readable message.

Status Code	Message	Notes
`ELQ-00001`	Total records processed.	
`ELQ-00002`	Invalid email addresses.	
`ELQ-00003`	Total records remaining after duplicates are rejected.	
`ELQ-00004`	Contacts created.	
`ELQ-00005`	Accounts created.	
`ELQ-00007`	CustomObject data created.	
`ELQ-00011`	Duplicate email addresses.	
`ELQ-00012`	CustomObject data updated.	
`ELQ-00013`	Contacts deleted.	
`ELQ-00014`	Accounts deleted.	
`ELQ-00016`	Contacts email marked as bounced.	
`ELQ-00017`	Contacts email marked as unsubscribed.	
`ELQ-00022`	Contacts updated.	
`ELQ-00023`	Accounts updated.	
`ELQ-00025`	CustomObject data linked.	
`ELQ-00026`	Duplicate identifier.	
`ELQ-00032`	Malformed records.	

Status Code	Message	Notes
`ELQ-00033`	Email addresses created.	
`ELQ-00034`	Total unmatched records.	
`ELQ-00035`	Email addresses updated.	
`ELQ-00048`	Contact already exists.	
`ELQ-00049`	Total new records.	
`ELQ-00055`	Campaign not found.	
`ELQ-00059`	Empty records.	
`ELQ-00060`	Total rejected records.	
`ELQ-00061`	Rejected records (missing fields).	
`ELQ-00062`	Duplicate account linkage.	
`ELQ-00067`	CustomObject data deleted.	
`ELQ-00068`	Assets created.	
`ELQ-00069`	Activities created.	
`ELQ-00070`	Activity Type not found.	
`ELQ-00072`	Asset Type not found.	
`ELQ-00077`	Event data created.	
`ELQ-00078`	Event data linked.	
`ELQ-00079`	Event data updated.	
`ELQ-00080`	Event data deleted.	
`ELQ-00081`	Event data rejected.	
`ELQ-00082`	Added members to program canvas step {stepId}.	
`ELQ-00084`	Opportunity does not exist.	
`ELQ-00085`	Contact does not exist.	
`ELQ-00101`	Sync processed for sync {syncId}, resulting in {status} status.	

Status Code	Message	Notes
`ELQ-00102`	Successfully exported members to csv file.	
`ELQ-00103`	Membership snapshot for export.	
`ELQ-00104`	Successfully converted file {file} ({kbUsed} kb) with {converter} converter.	
`ELQ-00105`	Error processing import staging file {file} on sync {syncId} for user {userId}.	Delete data from staging area for the definition, re-upload data, and sync. If the same error occurs again you can submit an SR to investigate.
`ELQ-00106`	Successfully converted csv file to sqlite, taking {kbUsed} kb.	
`ELQ-00107`	There was an error processing the {type}.	Delete data from staging area for the definition, re-upload data, and sync. If the same error occurs again you can submit an SR to investigate.
`ELQ-00109`:	Unable to convert file {file} with {converter} converter. File will remain locked and will NOT be processed.	
`ELQ-00110`	Trying to unlock file {file} on import {importId} for user {userId}. File has been {lockedState}.	
`ELQ-00111`	Field {field} is not part of import definition, and will be ignored.	
`ELQ-00112`	`identifierFieldName`	

Status Code	Message	Notes
	{field} must be contained within Fields, if specified.	
`ELQ-00114`	Added members to Account List {listId}.	
`ELQ-00115`	Subscribed members to Email Group {groupId}.	
`ELQ-00116`	Unsubscribed members from Email Group {groupId}.	
`ELQ-00117`	Removed members from Account List {listId}.	
`ELQ-00118`	Removed members from Contact List {listId}.	
`ELQ-00119`	Added members to Contact List {listId}.	
`ELQ-00120`	Updated members status to {status} in Cloud Connector Instance {instanceId}.	
`ELQ-00121`	Field {field} is part of import definition, but not included in file {file}.	
`ELQ-00122`	Validation errors found on sync {syncId} for user {userId}. Sync will NOT proceed.	This indicates there is a validation error with the definition, for which more details on the validation error are provided in the log message. Restart from the beginning with creating a new valid definition.
`ELQ-00123`	`identifierField` will be ignored since	

Status Code	Message	Notes
	Activities are never updated.	
`ELQ-00124`	There was no data available to process.	
`ELQ-00125`	Constraint failed.	
`ELQ-00126`	There are no fields to process in the file {file}.	
`ELQ-00127`	There are no mapped fields in the file {file}.	
`ELQ-00128`	Updated members status to {status} in service instance {instanceML}.	
`ELQ-00129`	Initialized internal staging for data transfer.	
`ELQ-00130`	Total records staged for import.	
`ELQ-00131`	Deleted internal staging for data transfer.	
`ELQ-00132`	Campaign not active.	
`ELQ-00133`	Unable to lock file.	
`ELQ-00134`	Successfully locked file.	
`ELQ-00135`	Multiple matched record limit exceeded.	
`ELQ-00136`	Staging area storage limit exceeded.	
`ELQ-00137`	Ready for data import processing.	
`ELQ-00138`	Contacts email marked as subscribed.	
`ELQ-00139`	Error staging data for sync actions.	Delete data from staging area for the definition, re-upload data, and sync. If the same error occurs again

Status Code	Message	Notes
		you can submit an SR to investigate.
`ELQ-00140`	Records staged for sync actions.	This is an informational message indicating the number of records staged.
`ELQ-00141`	Error when exporting members to csv file.	
`ELQ-00142`	Error when exporting members to csv file. Retrying.	
`ELQ-00143`	Sync Recovery Worker could not recover from sync error. {syncProductType} definition for id {syncReferenceId} has been deleted.	This indicates the definition was deleted before the sync completed. Restart from the beginning and be sure to not delete the definition before the sync completes.
`ELQ-00144`	Total records with rejected fields.	
`ELQ-00145`	The request failed and will be retried automatically in the specified time.	
`ELQ-00146`	The sync failed because of a temporary error.	Retry the sync after an hour. If the same error occurs again you can submit an SR to investigate.

HTTP status codes

In most cases Eloqua's APIs will respond to requests with the standard HTTP status code definitions. If the API fails to validate a request, it will respond with a validation error message (JSON or XML) describing the issue.

Here are some common HTTP status codes you may encounter:

- 200: "OK"
- 201: "Success"
- 204: "Success"
- 301: "Login required"
- 304: "Not Modified"
- 400: "There was a parsing error."
- 400: "Bad Request"
- 400: "There was a missing reference."
- 400: "There was a serialization error."
- 400: "There was a validation error" (see Validation errors)
- 401: "Login required"
- 401: "Unauthorized"
- 401: "You are not authorized to make this request"
- 403: "Forbidden"
- 403: "This service has not been enabled for your site."
- 403: "XSRF Protection Failure"
- 403: "Eloqua Analyzer license is required"
- 404: "The requested resource was not found."
- 409: "There was a conflict."
- 412: "The resource you are attempting to delete has dependencies, and cannot be deleted"
- 413: "Storage space exceeded."
- 429: "Too Many Requests"

- 500: "The service has encountered an error."
- 500: "Internal Server Error"
- 502: "Bad Gateway"
- 503: "Service Unavailable".
- 503: "There was a timeout processing the request".

> **Note**: 503 status codes indicate a temporary server error that is likely to be alleviated after some delay. It's recommended to check Eloqua System Status, and wait to retry after any planned or unplanned unavailability is complete. If Eloqua is available at the time of the 503, it's recommended to retry, placing more delay between retries if the requests continue to be unsuccessful.

Validation errors

If the API fails to validate a request, it will respond with a 400 validation error message (JSON or XML) describing the issue.

The below validation errors are the most common and will respond with some details, including a list of validation messages. Validation errors typically occur when a request is malformed -- usually because a field has not been given the correct value type, or the JSON is misformatted.

Error types

- **EndpointParameterError**: caused by incorrect input parameters
- **BooleanRequirement**: the value must be either "true" or "false"

- **ContentRequirement**: the content supplied is blacklisted
- **DateRequirement**: the value must be a proper date
- **IdRequirement**: the value must be an integer greater than 0
- **ImmutabilityRequirement**: the value cannot be changed once it has been set
- **IntegerRequirement**: the value should be an integer
- **NoDuplicatesRequirement**: there is a duplicate value supplied in the request
- **NotNullRequirement**: the value cannot be null or missing
- **ObjectValidationError**: a problem with one of the properties on the object

  Some examples include:

  - **TextWithNoUrlRequirement**: the value must not contain URLs
  - **TextWithNoHtmlRequirement**: the value must not contain HTML
  - **DateRequirement**: the value must contain a date
  - **EmailAddressRequirement**: the value must contain an email address
  - **ValidTextLengthRequirement**: the value does not contain a valid number of characters
- **RelationshipRequirement**: a relationship between two objects or related properties has not been met
- **SyntaxRequirement**: the HTML is not valid or could not be parsed
- **UniquenessRequirement**: the supplied name is not unique
- **UrlRequirement**: the supplied value should be a valid url

Example

For example, if you try to update a contact with id "sandwich" instead of an integer greater than zero:

```
GET https://.../data/contact/sandwich
Accept: application/json
```

You will receive a 400 response with "IdRequirement" error:

```
[
  {
    "type":"EndpointParameterError",
    "parameter":"id",
    "requirement":{
      "type":"IdRequirement",
      "isRelativeAllowed":false
    },
    "value":"sandwich"
  }
]
```

From the above "Error types" list, we see that this error means " the value must be an integer greater than 0".

Determining base URLs

Eloqua supports multiple data centers, each with a unique domain name. Eloqua refers to these as "pods", there are currently seven in total - p01, p02, p03, p04, p06, p07, and p08. The `https://login.eloqua.com/id` endpoint allows you to programmatically discover a given Eloqua instance's pod, associated domain name, and base URL, which is required for making API calls to the instance.

Validate your base URL before making any API calls. If you do not validate, your API calls may not work. New Eloqua users may be added to different data centers, and some Eloqua instances may periodically move between data centers. For any application that will be built and used by many customers, base URL discovery should

be built into the your app. While not common, there are cases where an instance's base URL will change (moving an instance between pods).

> **Note**: To call `https://login.eloqua.com/id` you need, at a minimum, the *Advanced Users - Marketing* permissions for your Eloqua instance.

To determine your base URL:

1. Send an authenticated request (in this example we use basic authentication, but OAuth2 is generally recommended as a more secure option):

   GET https://login.eloqua.com/id

   Authorization: Basic XXXXX...

   For more details, see basic authentication.

2. If you receive a **success** response, the response body will look like:

```
{
  "site": {
    "id": 42,
    "name": "MyTestInstall"
```

```
  },
  "user": {
    "id": 314,
    "username": "jane.smith",
    "displayName": "Jane Smith",
    "firstName": "Jane",
    "lastName": "Smith",
    "emailAddress": "jane.smith@eloqua.com"
  },
  "urls": {
    "base": "https://secure.p03.eloqua.com",
    "apis": {
      "soap": {
        "standard": "https://secure.p03.eloqua.com/API/{version}/Service.svc",
        "dataTransfer": "https://secure.p03.eloqua.com/API/
{version}/DataTransferService.svc",
        "email": "https://secure.p03.com/API/{version}/EmailService.svc",
        "externalAction": "https://secure.p03.eloqua.com/API/
{version}/ExternalActionService.svc"
      },
      "rest": {
        "standard": "https://secure.p03.eloqua.com/API/REST/{version}/",
        "bulk": "https://secure.p03.eloqua.com/API/Bulk/{version}/"
      }
    }
  }
}
```


If you receive a **failure** response, the response body will look like:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 20

"Not authenticated."
```

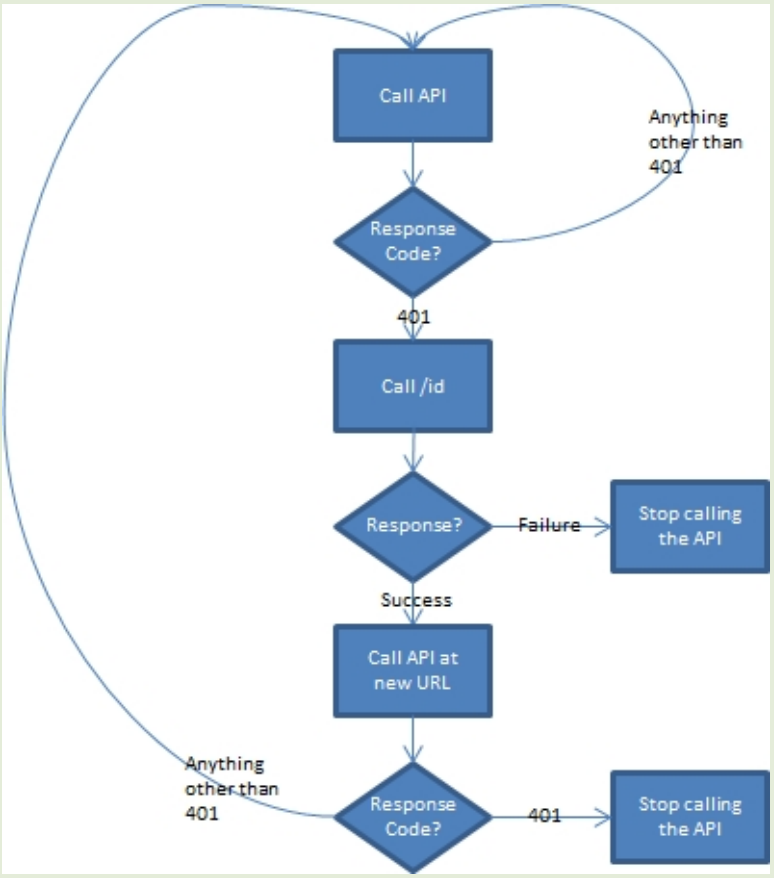
3. Now that you have the base URL, you can use it in your code. In the above example, you would use `https://www05.secure.eloqua.com` as the base URL for any API calls you make to Eloqua.

4. Cache the data from the `/id` endpoint for the duration of the user's session or for a reasonable period of time.

> ❑ **Warning**: Do not call the `/id` endpoint each time you make an API call. There will be throttling or rate-limiting imposed on the `/id` endpoint to prevent this behavior. Instead, call it once per session (or after a reasonable period of time) and cache the results.

> **Note**: When calling this endpoint from a Cloud Connector or Cloud Component app, if an API call fails with a 401, your application should call the `/id` endpoint again to determine whether the 401 was the result of the user's credentials/token being invalidated, or the result of the site being moved to a different data center. If the `/id` endpoint returns a failure, your application should stop making API calls. If the `/id` endpoint returns a success, your application should retry the API call at the new

address in the response. Here's a flowchart showing the fallback process:



Interactive applications can respond to 401s using this process, or by simply logging the user out.

Sending API requests using cURL

Client for URLs (or cURL) is a software project comprised of two development efforts - cURL and libcurl.

libcurl is a free, client-side URL transfer library with support for a wide range of protocols. libcurl is portable, thread-safe, feature rich, and well supported on virtually

any platform. It is probably the most popular C-based, multi-platform file transfer library in use.

cURL is a command-line tool for getting or sending files using URL syntax. Since cURL uses libcurl, it supports the same range of common Internet protocols that libcurl does.

This tutorial will walk through the format and syntax used when making Oracle Eloqua API requests with cURL. If you are not yet familiar with the structure of requests, see API requests.

In this tutorial:

- Authentication
- Sending a GET request
- Sending a DELETE request
- Sending a POST request
- Sending a PUT request

> **Note**: The code samples in this tutorial are developed for bash. See cURL request formatting for more information on how to structure cURL requests for other command line tools.

Authentication

The easiest way to authenticate with Oracle Eloqua's APIs is to use basic authentication which uses your Eloqua company name, user name, and password to

authenticate. To use basic authentication, use the cURL `--user` option followed by your company name and user name as the value. cURL will then prompt you for your password.

```
curl --user "<companyName>\<userName>" --request GET
https://secure.p0<podNumber>.eloqua.com/api/<apiType>/<apiVersion>/<endpoint>

Enter host password for user '<companyName>\<userName>':
```

To avoid the password prompt, append your password after your user name by following the syntax `--user "<companyName>\<userName>:<password>"`, but this will leave your password in shell history and isn't recommended.

> **Note**: For this tutorial, the company name we use is `APITest`, our user name is `API.User`, and our pod is `3`.

Sending a GET request

Let's retrieve the first two contacts within your database. Only the `--user` option is required. The `--request` option is optional, but we include it in the example below for conformity with other requests to establish a pattern.

```
curl --user "APITest\API.User" --request GET
https://secure.p03.eloqua.com/api/REST/1.0/data/contacts?count=2

{
   "elements":[
      {
         "type":"Contact",
         "id":"1",
```

```
            "createdAt":"1403034086",
            "depth":"minimal",
            "name":"george.washington@america.com",
            "updatedAt":"1410193024",
            "emailAddress":"george.washington@america.com"
        },
        {
            "type":"Contact",
            "id":"2",
            "createdAt":"1403113589",
            "depth":"minimal",
            "name":"john.a.macdonald@canada.com",
            "updatedAt":"1403113589",
            "emailAddress":"john.a.macdonald@canada.com"
        }
    ],
    "page":1,
    "pageSize":2,
    "total":527
}
```


Sending a DELETE request

For `DELETE` requests, the `--user` option and `--request` option are both required. Let's delete a contact whose contact id is `1`.

```
curl --user "APITest\API.User" --request DELETE
https://secure.p03.eloqua.com/api/REST/1.0/data/contact/1

200 OK
```

Note that for `DELETE` requests, there is no body is returned, just a response status code.

Sending a POST request

`POST` requests are a little different. Let's look at an example of how to create a contact whose email address is "george.washington@america.com".

```
curl --user "APITest\API.User" --header "Content-Type: application/json" --request POST --data '{"emailAddress":"george.washington@america.com"}' https://secure.p03.eloqua.com/api/REST/1.0/data/contact

{
   "type": "Contact",
   "currentStatus": "Awaiting action",
   "id": "1",
   "name": "george.washington@america.com",
   "emailAddress": "george.washington@america.com",
   ...
}
```

As you can see `POST` requests are much different than the other verbs we've looked at so far, so let's take a closer look at what's happening in the request.

- The `--header` option is required with a `Content-Type`. All `POST` and `PUT` requests to the Application and Cloud APIs should be `application/json`, while the Bulk API supports `application/json` and `text/csv`. To confirm the media types supported, see the "Supported Media Types" per endpoint in our endpoint documentation.
- The `--data` option is optional, but we include it for readability.

Sending a PUT request

`PUT` requests follow the same format as `POST` requests. Let's take a look at an example of how to update a contact to change the contact's business phone number when the contact `id` is `1` and email address is "george.washington@america.com".

```
curl --user "APITest\API.User" --header "Content-Type: application/json" --request PUT --data '{"id":"1","emailAddress":"george.washington@america.com","businessPhone":"555-555-5555"}' https://secure.p03.eloqua.com/api/REST/1.0/data/contact/1

{
  "type": "Contact",
  "currentStatus": "Awaiting action",
```

```
  "id": 1,
  "name": "george.washington@america.com",
  "emailAddress": "george.washington@america.com"
  ...
}
```

cURL request formatting

Depending on the command line tool you use, the format in which you use cURL to make API requests will differ. See your command line tool below for the cURL syntax to use when making requests to Oracle Eloqua's APIs.

- Bash
- Windows command line
- Terminal (macOS)

Bash

When using bash command line, format your cURL requests according to the following guidelines.

cURL option	Syntax	Example
--user	"<instanceName>\<userName>"	"APITest\API.User"
--data	'{"<key>":"<value>"}'	'{"emailAddress":"george.washington@america.com"}'

Syntax

```
curl --user "<instanceName>\<userName>" --header "Content-Type: application/json" --request POST --data '{"<key>":"<value>"}' https://secure.p0<podNumber>.eloqua.com/api/<apiType>/<apiVersion>/<endpoint>
```

Example

```
curl --user "APITest\API.User" --header "Content-Type: application/json" --request POST --data '{"emailAddress":"george.washington@america.com"}' https://secure.p03.eloqua.com/api/REST/1.0/data/contact
```

Windows command line

When using Windows command line, format your cURL requests according to the following guidelines. Note that JSON must be serialized differently in windows command line compared to other command lines.

cURL option	Syntax	Example
--user	"<instanceName>\<userName>"	"APITest\API.User"
--data	"{\"<key>\":\"<value>\"}"	"{\"emailAddress\":\"george.washington@america.com\"}"

Syntax

```
curl --user "<instanceName>\<userName>" --header "Content-Type: application/json" --request POST --data "{\"<key>\":\"<value>\"}" https://secure.p0<podNumber>.eloqua.com/api/<apiType>/<apiVersion>/<endpoint>
```

Example

```
curl --user "APITest\API.User" --header "Content-Type: application/json" --request POST --data "{\"emailAddress\":\"george.washington@america.com\"}" https://secure.p03.eloqua.com/api/REST/1.0/data/contact
```

Terminal (macOS)

When using terminal, format your cURL requests according to the following guidelines.

cURL option	Syntax	Example
--user	"<instanceName>\\<userName>"	"APITest\\API.User"
--data	'{"<key>":"<value>"}'	'{"emailAddress":"george.washington@america.com"}'

Syntax

```
curl --user "<instanceName>\\<userName>" --header "Content-Type: application/json" --request POST --data '{"<key>":"<value>"}' https://secure.p0<podNumber>.eloqua.com/api/<apiType>/<apiVersion>/<endpoint>
```

Example

```
curl --user "APITest\\API.User" --header "Content-Type: application/json" --request POST --data '{"emailAddress":"george.washington@america.com"}' https://secure.p03.eloqua.com/api/REST/1.0/data/contact
```

App developer frequently asked questions

Why should I develop apps for the app developer framework?

The app developer framework is a complete development platform where you can easily build, register, and deploy apps for Eloqua. With new and improved service types which take advantage of Eloqua's bulk API, support for OAuth, and the ability to test your applications with Eloqua prior to launch, the App Developer Framework provides the environment needed to put apps first.

What permissions do I need to start developing apps for the Oracle Eloqua app?

At the minimum, you need access to an Eloqua instance. If you are not currently an Eloqua user, you can sign up as a technology partner to obtain a development instance. As a user, you will also need the **Advanced Users - Marketing** permissions.

When and how does my app get listed on the Oracle Eloqua app site?

See building apps for the Oracle marketing app developer framework on Topliners for detailed instructions.

How do marketers find my app to start using it?

Registered apps are listed on the Oracle Cloud Marketplace. Users are linked to this page through the "Get more Apps" link in Eloqua's **AppCloud Catalog** section.

No. The Oracle Eloqua app exclusively supports contact, account, activity and custom object fields in its record definition fields. You must use the correct Eloqua markup language statement to reference each field.

Is there an Oracle Eloqua app certification program?

Yes. Check out the Oracle Marketing AppCloud Certification Program on Topliners.

What kind of digital certificate is required to communicate with Eloqua?

App's URL endpoints must have an SSL Certificate, specifically needing to be a digital signature from a certificate authority (CA). Eloqua will not communicate with untrusted sites.

What happens during an Eloqua release?

The application and the various associated services will be intermittently unavailable within the duration of the maintenance window. If there are any Bulk API sync failures, they should be retried after the maintenance window completes.

Which version of TLS is supported?

TLS 1.2 is supported.

Oracle Eloqua App Services and Operations

Which service should I develop?

It depends on what you're trying to achieve! See the Service Descriptions for an overview of each service and the use cases it supports.

What's the difference between Oracle Eloqua App services and Cloud Connectors or Components?

Eloqua Oracle Eloqua app services greatly extend the functionality provided by cloud connectors and cloud components. Cloud content replaces cloud components, allowing you to process emails in bulk and test the service within Eloqua. Unlike cloud connectors and components, the Oracle Eloqua App services use Eloqua's bulk API for processing, greatly improving performance and throughput.

Can I include campaign, email, landing page, form fields in my record definition?

No. The Oracle Eloqua app exclusively supports contact, account, activity and custom object fields in its record definition fields. You must use the correct Eloqua markup language statement to reference each field.

Can I include static values in my record definition?

No. You can only specify the Eloqua markup language for Eloqua fields.

Managing Your Apps

What does this app status mean?

See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

What if my request to refresh a token times out?

If the current access token has not been used, submitting a request to authenticate with the previous refresh token will return the existing new access token and refresh token.

What happens if contacts encounter an error in an Action or Decision step? What happens if the service is unavailable when contacts flow into a step on the canvas?

The contacts remain in the step until the marketer manually pushes them into the next step on the canvas.

Does Eloqua notify me when someone uninstalls or deletes my app from their Eloqua instance?

If you set an Uninstall URL it will be called when a user uninstalls the app.

Why has my app been shut down?

If in the last five minutes, there were more than 100 calls, and 25% of them failed, the app will be shut down. All the pending and future requests will be marked as failed. See the App shutdown for more information.

Will Eloqua retry my contacts if my app doesn't respond?

No, the contacts in a step will be marked as "Errored". If the marketer has configured a default path for the contacts to flow through, then the contacts will flow into the next step.

What if my app responds to the notify call after 100 seconds?

Any response after 100 seconds will be ignored.

What if my app responds to the notify call with a response status code that is not 200 level?

If Eloqua calls out to your app and receives a response status code that is between 300 and 599, Eloqua will retry the notify call over approximately an eight-hour period of time with a backoff strategy of the time between calls doubling after each call. After

this eight-hour period, the contacts in a step will be marked as "Errored". If the marketer has configured a default path for the contacts to flow through, then the contacts will flow into the next step.

Why does my app prompt the error "502 Error: App Provider not Available"?

This error message will appear if a URL has been configured with a non-standard port. Eloqua only supports the standard ports 80 and 443. If a URL is not configured with port 80 or port 443, requests to this URL will fail.

Limits

Are there limits that I should be aware of?

The AppCloud developer framework relies on the bulk API. The bulk API has limits on the size of the staging area for imports and exports, on the amount of data you can import at one time, and on the number of fields you can include in a record definition. There is also a daily limit on the number of syncs you can perform.

What happens if I reach the daily sync limits?

The daily sync limit is not currently enforced, but syncs are logged and monitored.

Bulk API frequently asked questions

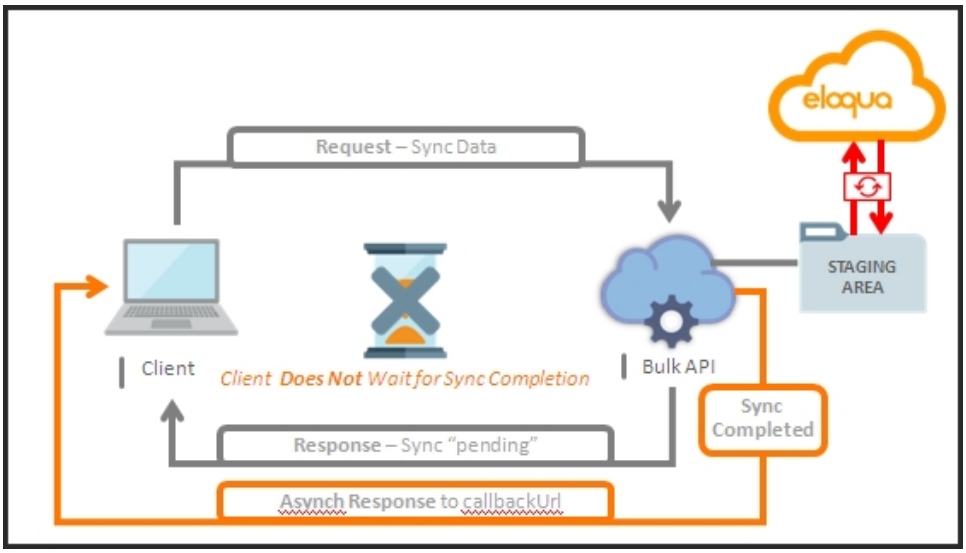
General

What trims/versions of Eloqua is the bulk API available?

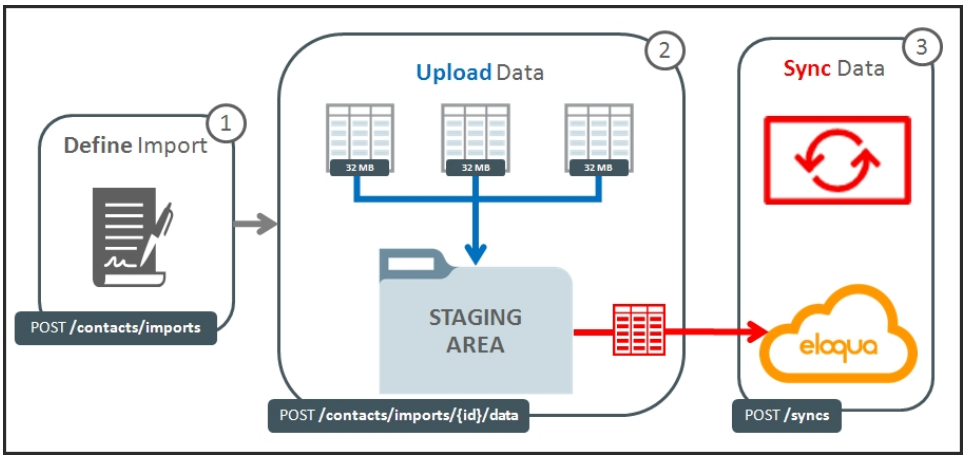
The bulk API is available on ALL trims and ALL versions of the API (E9 and E10, and Express, Team and Enterprise)

I am having trouble visualizing the Bulk API pattern? Are there any visual diagrams?

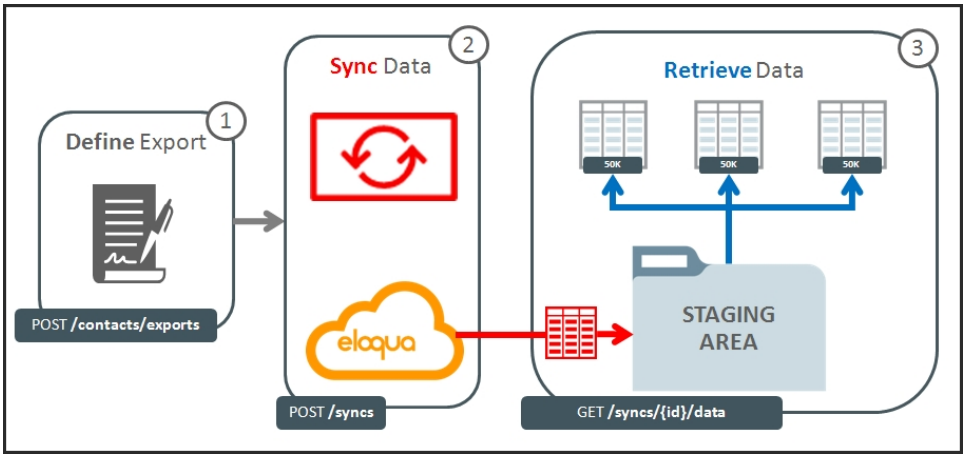
Asynchronous Flow



Importing Data



Exporting Data



Is the API rate limited?

Every Eloqua instance has the following soft limits by Bulk API sync type:

Bulk API Sync Type	Limit
Export	2000 per hour
Import	2000 per hour
Sync Action	4000 per hour

Here are the Bulk API limits:

- The amount of data allowed on our servers in the staging area
    - The amount is based on the # of contacts allowed for the client x 50 in KB; So, for a client with 10 M contacts, this would result in 500 GB storage area for that client
- Import Limits
    - Definition limit: An import definition can include a maximum of 100 fields
    - Request limit: There is a hard limit of 32 MB per request when posting data to an import sync's staging area

        - Staging limit: It is recommended to sync an import whenever approximately 250 MB of data has been posted to its staging area

- Export Limits
    - Definition limits: An export definition can include a maximum of 250 fields
    - Request limit: There is a limit of 50,000 records per request while retrieving data from an export
    - Staging limit: When exporting activities there is a limit of 5 million records per sync.

Read more about limits.

Does the bulk API provide the ability to export contact data of any given set of fields based on some filter or other requirement?

Yes; an export is limited to 250 fields at once, but any of the contact fields are available at export as are the linked account fields. In addition, Contact attributes (subscription and bounceback status) are available.

For filtering, data can be exported from a contact list, filter or segment, or based on un/subscription to an email group, a cloud connector status or comparison of a single field on the contact. See Filtering for more information.

How do you retrieve all contacts that are in a contact list?

Export contacts based on their presence in the contact List with ID 123:

```
{
  "name": "List 123 Contacts Export",
  "fields": {
    "ID": "Contact.Id",
```

```
    "FirstName": "Contact.Field(C_FirstName)",
    "LastName": "Contact.Field(C_LastName)",
    "EmailAddress": "Contact.Field(C_EmailAddress)"
  },
  "filter": "EXISTS('ContactList[123]')"
}
```

See Filtering for more information on filtering.

Can you import a bunch of contacts with an arbitrary set of fields?

Yes, imports are also defined with any set of contact fields / attributes as are exports.

Can you import into a (contact) group/list?

Yes; similar to export filtering, except that you specify additions to a contact list by using a Sync Action, with the action = "add" and the `destinationUri` as the contact list's uri (ie "/contact/list/123"). See for Sync actions more information.

Can you import into a canvas?

You can create a Feeder Service to enable importing directly into a canvas.

Another option for importing contacts into a canvas is using a Bulk API sync action to add to list. The contact list could be used in a segment that gets re-evaluated which then would add them to a canvas.

Can you unsubscribe contacts from all future mailings (when importing)?

There are two options here:

Set the contact's global subscription status to unsubscribed. You interact with this almost like any other field (ie as a column in the import).

On import, use a sync action to unsubscribe from a particular email group.

What is the maximum number of fields that can be included in an export?

250

> **Note**: When specifying fields for an app or service instance, you can specify a maximum amount of **249** fields because AppCloud developer reserves the contact ID field, which is required to update the status of records and move them to subsequent steps or pass cloud content.

I am coming close to the Bulk API syncs soft limit, or am already over this limit for one or more sync types. Am I going to get shut down? What's the impact of going over?

We want to encourage app/Bulk API usage, as long as app/Bulk API usage is not generating abusive API behavior. We advise not getting too caught up with the soft limit for Bulk API syncs, unless going over significantly, and/or observing performance issues related to the Bulk API load.

If you are going over this limit for one or more sync types, the chances of performance issues will increase with volume (meaning the higher you exceed the limit, the higher probability of experiencing performance issues). It's strongly recommended to stay as

close to the soft limits as possible, to reduce the chance of performance issues. The main impact you'll see as the sync volume increases - is increased sync processing time with increased probability of performance issues. Performance issues are more likely to occur when there is a large Bulk API sync volume with other Eloqua activities (such as Lead Scoring, Form Processing, Segment Execution, and so on) occurring concurrently and/or also in large volumes. The lower the Bulk API sync volume, the smaller probability of performance issues when these other activities are occurring.

I am exporting all of my activities using the Bulk API. When I accumulate activities exported via the Bulk API, metrics are not matching what I see an Insight. Why wouldn't they match?

The are two primary reasons they don't match:

1. Any data pulled from APIs does not include activity from deleted Contacts. Insight includes activity from deleted Contacts.

2. Label-based Access Control (LBAC), if it's being used. The APIs respect LBAC, so sometimes the user placing the API request doesn't have access to view Contacts, or the User doesn't have access to view Contacts in Insight. Additionally, non contact related reports in Insight are not limited to the contacts that a user can access. For example, a campaign report will show data related to all campaign members.

Are there any best practices or resources for exporting all activities via the Bulk API?

Exporting All Eloqua Activities Using the Bulk API: Flowchart, Best Practices, and Resources

Is it possible the Bulk API would export duplicate records?

Yes, if you use `UpdatedAt` in the definition's filter the same record may be returned more than once if a record is updated during the export. This applies to all Bulk API exports that allow `UpdatedAt` in the filter.

Oracle Eloqua

Developer Guide

Contents

App Developer Framework

The Oracle Eloqua AppCloud developer framework is a complete development platform where you can easily build, register, and deploy apps for Eloqua. With new and improved service types which take advantage of Eloqua's bulk API, support for OAuth, and the ability to test your applications with Eloqua prior to launch, the Oracle Eloqua app developer framework provides the environment needed to put apps first.

The framework allows third parties to register and manage the integrations that they build for Eloqua right from their own Eloqua development instance. It also offers common, repeatable patterns for building apps that extend the functionality of Eloqua, providing a consistent, seamless experience for the marketing user.

Features

- App keys and OAuth2
- Test apps inside your development instance before going live
- Selective whitelististing so you can run betas on your apps before release
- Improved processing speeds with the bulk API
- A seamless flow for app installation and configuration
- 6 Oracle Eloqua app service types to support your different goals. For more details, see service types.

Flow

To get started with the Oracle Eloqua app developer framework:

1. Register as a provider in your Eloqua development instance. This gives you provider credentials that you can use to develop apps that interact with Eloqua.
2. Develop an app.
3. Register your app with Eloqua by providing a series of *templated URLs* that Eloqua uses to call out to your app. See the register services guide for details about what URL parameters you should include in each templated URL so that you will receive the data you need to do whatever your app does.
4. Publish the app to specific Eloqua instances or as a general release.
5. Whitelist a site and share your app's install URL so that the site's administrator can install the app and let marketers start using it.

Get started with the App Developer Framework

The Oracle Eloqua app developer framework is designed to make it easy to start working with the framework with minimal setup effort required. Once you have registered as a provider and ensured that your Eloqua user has sufficient permissions, you are ready to start developing an app!

Requirements

- An Eloqua instance in which you have registered as a provider.

> **Note**: If you are an external partner and want to publish your app on the Oracle CX Marketplace, see Becoming a Partner and Planning App Development to learn how to submit your app proposal before requesting an Eloqua Sandbox.

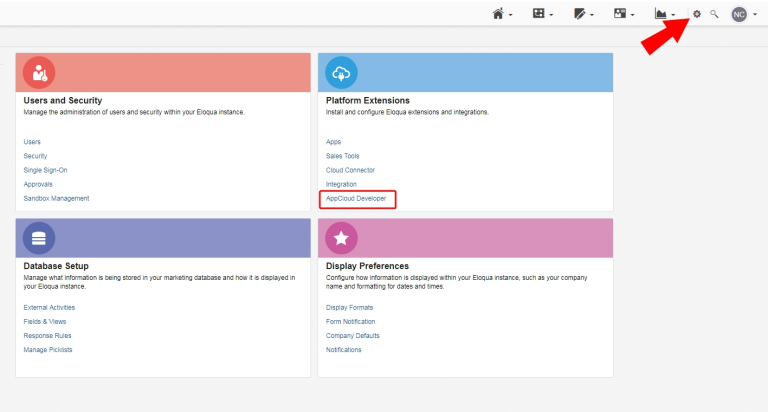
- Sufficient user permissions: the user making calls against Eloqua's APIs must have **Advanced Users - Marketing** permissions granted in the security groups section of the user management area of Eloqua.

Creating a provider

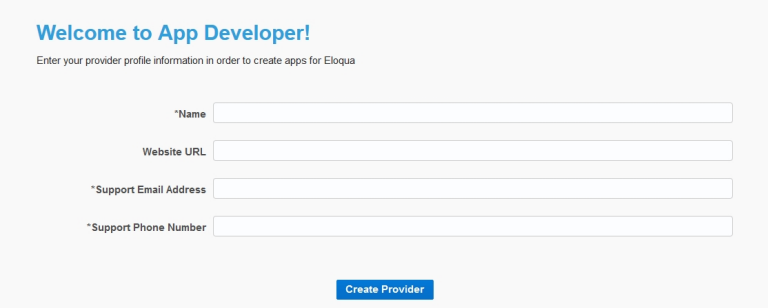
Learn more by watching the video

Registering as a provider

To register as a provider, log in to your Eloqua instance, and click **Settings** > **AppCloud Developer**.



Enter your company's information in the welcome page:

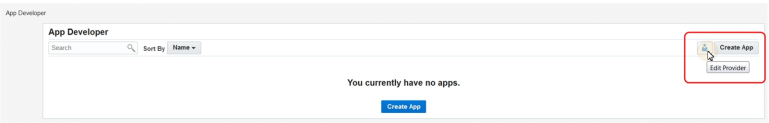


- **Name**: Your provider name. You might want to use your company name.
- **Website URL**: Your company website. This is optional.
- **Support email address**: The email address users should use to contact you if there are issues with your app.
- **Support phone number**: A phone number where you can be contacted if there are problems with your app.

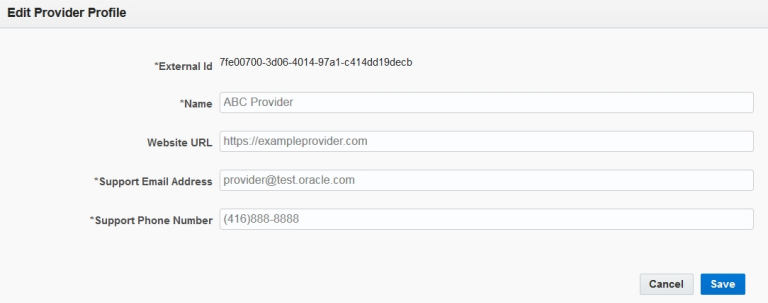
Click **Create Provider** > **Save**, and you're all set up as a provider. Your unique GUID-based provider **External ID** is displayed.

Edit provider information

You can edit this information by clicking **Edit Provider**.



The **Edit Provider Profile** page lists your provider information.



Next steps

Once you register as a provider, read the service descriptions to determine which ones you want to implement in your app, read about URL templating, and then consider the develop an app.

Becoming a Partner and Planning App Development

This topic explains how to become an Oracle Partner Network (OPN) member to start publishing apps on the Oracle Cloud Marketplace. On this page, *External partners* are defined as entities outside of Oracle who want to develop apps for the Eloqua App Framework.

The steps to plan an app are:

1. Evaluation & Onboarding
2. Access Developer Resources
3. Build and Test Your App
4. Review Your App
5. Release Your App

Step 1 - Evaluation & Onboarding

a. Submit the details of your proposed app using the evaluation form. If approved, the Eloqua Apps team will advise you to continue with the process.

b. Join the Oracle Partner Network. Applicable only to external partners. Oracle entities interested in developing generic apps for Eloqua on the App Framework can ignore this step.

> **Note**: Make sure to accept the Oracle CIA (Oracle Cloud Interoperability Appendum). This is a prerequsite for getting sandbox account access.

Step 2 - Access Developer Resources

a. Request access to an Eloqua Sandbox account (applicable only if you do not have your own Eloqua instance and you are an external partner who has already joined OPN and signed the CIA)

b. Join the Eloqua AppCloud Insiders Developer Community.

c. Access other useful developer resources:

- App Developer Framework Documentation
- Getting Started Videos
- Which Eloqua API Should I Use?
- Authentication
- Determining Base URLs
- Eloqua API Tutorials

- App Developer Framework Overview
- Changelog

Step 3 - Build and Test Your App

a. You are now ready to build your app. Make use of all the developer resources provided in the Eloqua Developer Help Center.

b. For any questions during app development, post them on the Eloqua AppCloud Insiders to get help from our tech experts and community members.

c. After you have developed your app and are ready to test, you must first Register your App with Eloqua.

Notes:

- The infrastructure for hosting the app is the responsibility of the App Provider.
- The implementation of the Framework services are optional.
- The use of OAuth Authentication to Eloqua APIs is required.

Step 4 - Review Your App

Before your submit your app for review, make sure you have thoroughly completed functional testing of your app.

a. Next, complete and send the Compatibility Review Document to the Eloqua Apps team. Please make sure you have taken care of all aspects covered in the document. As part of the review process, the Eloqua Apps team may ask for additional information and/or a demonstration of the app.

Step 5 - Release Your App

Step 5.1 Prepare App Resources

Prepare the resources for your app's release such as:

- Detailed user guide
- Pitch deck, demo videos etc.
- Customer on-boarding process
- Support model
- Enabling Oracle teams to showcase the app

Step 5.2 Publish Your App on the Marketplace

The Oracle Cloud Marketplace offers a platform for software vendors to show case their software that inter-operates with and extends Oracle CX Cloud (SaaS) Applications.

To publish your App on the Marketplace:

a. Become a Cloud Marketplace Publisher.
   - You must be an OPN member with an Oracle.com login. If you are an OPN member, but you do not have an Oracle.com login, follow the New User registration to associate your Oracle.com login with your OPN Membership Company ID.
   - Ensure to select only the Oracle products you are providing an integrated application.

b. Create a listing on the Cloud Marketplace following the Oracle Cloud Marketplace Listing Guidelines.

i. Sign in to the Oracle Cloud Marketplace Partner Portal

ii. Click **Home > Create Listing > Application Listing**.

iii. In the **Demo URL** field, enter a video demonstration of your app hosted on YouTube or Vimeo.

iv. In the **Additional Information** tab from your app listing, complete and submit the following documents from the **Get Templates** section:

- Security Questionnaire, Test Plan
- Integrates-with Supplement - <Oracle CX Cloud Product> (required if your app is hosted on the Oracle Cloud)
- Partner Supplied Template: Deployment Guide (describes how to install, setup, and configure your application with the Oracle CX Cloud product)

Your listing may be returned for further editing to improve overall content or add/correct content. Reference the Marketplace Guidelines document for requirements and tips.

c. Publish your Marketplace listing

After the listing, video demonstration and technical documentation are submitted, your listing will be reviewed and approved by Oracle. You will receive a notice that your listing has been approved to publish. Now you can publish your listing within the Partner Portal!

i. Select your listing under **Submitted** and click **Publish**.

Service descriptions

Action services

Actions allow a marketer to add an action step to a campaign or program that is performed by an external system. For example, an action service could send contacts SMS messages, trigger a direct mailing, or register a contacts for a webinar. When contacts enter an action step, Eloqua calls out to the app, and the app responds by performing an action in an external system. The app then informs Eloqua that the

process is complete and the contact moves on in the workflow. Read more about actions in the development guide and service registration instructions.

Learn more by watching the video

Decision services

Decisions allow a marketer to delegate a decision-making step in a campaign to an external system without having to bring the system's data into Eloqua. Decision services evaluate contacts using externally-held data, and determine whether the contacts meet the criteria. The service responds with yes or no to determine which path the contact should take through the campaign or program workflow. For example, a decision service could check if a contact exists in a CRM system, and sort the contacts accordingly. With Eloqua decision services, the data used in the decision never needs to be brought into Eloqua. Read more about decisions in the development guide and service registration instructions.

Learn more by watching the video

Feeder services

With feeders, developers can build apps that use data in third-party services to drive campaign and program membership. Feeders allow marketers to populate a campaign or program with contacts or custom objects from an external system, automatically importing the contacts or custom objects into Eloqua and adding them to a campaign or program. Read more about feeders in the development guide and service registration instructions.

Note: For marketers using Eloqua's campaign canvas, feeder services are referred to as AppCloud Audiences.

Learn more by watching the video

Content services

Content allows marketers to source pieces of content in Eloqua emails and landing pages from an external source. This is the new and improved version of Eloqua's cloud components framework, and it includes many improvements such as asynchronous bulk processing (for emails), the ability to fully test the content service within Eloqua, and design-time interaction with the email and landing page editors. Read more about content in the development guide and service registration instructions.

Menu services

The AppCloud menus service is all about *context*. AppCloud menus enable a marketer to launch an externally-hosted application from within Eloqua via a menu dock. This menu dock floats on the right side of the screen in the campaign canvas or asset editor screens. Read more about menus in the development guide and service registration instructions.

Firehose services

The firehose service acts as a web hook and notifies an external system when a marketer performs an action in Eloqua. For example, when a marketer creates a campaign or edits an email, Eloqua can call out to the third-party with the update.

Read more about the firehose service in the development guide and service registration instructions.

Introduction to URL templating

The Oracle Eloqua app developer framework supports *URL Templating* to enable you to configure the service URIs that Eloqua calls out to. Any acceptable template parameter is substituted with the appropriate value before the call is made.

Common URL parameters include `{UserId}`, the unique identifier for the user whose actions triggered the call, `{InstanceId}`, the GUID-based ID that identifies an instance of a service, and `{EventType}`, which provides campaign status information (created, activated, draft, and so on). These URL parameters enable you to configure the service URLs that Eloqua calls out to, specifying the information that you need for your application.

Example

Consider this templated URL:

```
https://awesomeapp.example.com/create/instance={InstanceId}&asset={AssetId}&site={SiteName}
```

When Eloqua calls the URL, the templated parameters are replaced with the appropriate values:

```
https://awesomeapp.example.com/create/instance=b5fc25ce-9709-42c4-a57d-caa00e23f658&asset=45&site=DocsExampleSite
```

Recommendations

Choose the URL parameters to include in your templated URLs based on your use case. These parameters provide contextual information: your application design will

determine which parameters you need. The full list of parameters is available. The URL parameters reference provides a description for each parameter. You should familiarize yourself with the supported URL parameters when designing your templated URLs.

The service registration guides also include suggestions for parameters to include for specific URLs. These recommendations are the minimum you will likely need. Include additional parameters if you require them.

Develop an app

The Oracle Eloqua app developer framework provides 6 service types that you can develop to extend Eloqua's functionality. Some services create steps that a marketer can add to the campaign canvas, or to an email or landing page, another service type enables marketers to launch third-party applications directly from within Eloqua's UI, while another yet creates a web hook that calls out to a third-party service when a marketer performs an action within Eloqua.

These services make it easy for developers to extend Eloqua's capabilities, providing new opportunities for marketers to harness both Eloqua data and external services to better serve their goals.

Before you start developing Oracle Eloqua app services, familiarize yourself with the instantiation-execution model, which actions, decisions, feeders, and content follow. Then, you can read the development guides:

- Develop an action service
- Develop a decision service

- Develop a feeder service
- Develop a content service
- Develop a firehose service
- Develop a menu service

Instantiation-execution model

With the exception of menus and the firehose, Oracle Eloqua app services follow an instantiation-execution model. When you add an decision or action step to a campaign in the campaign canvas, a new instance of that decision or Action service is created. When that campaign or program reaches the decision or action stage, the service is executed. Similarly, when content is added to an email or landing page, a new instance of that content service is created.

Service instantiation

1. A Marketer drags the service onto a campaign or program canvas or onto an email or landing page asset. This triggers a call from the Eloqua UI to an internal Eloqua API that will interact with your servers.
2. The internal Eloqua API calls out to your service using the templated create URL you configured when registering your app. This is a POST request, authenticated with OAuth, which contains an empty JSON object. The call provides you with the new instance's GUID.

   For example, AwesomeApp has an AppCloud Decision service whose Create URL is:

   ```
   https://example.com/awesomeapp/decide/create?instance={instanceId}
   ```

   Eloqua calls out to this URL with an empty JSON object, replacing the templated parameters with the service instance ID and app installation ID.

Your application should respond with default details in a Record Definition Data Transfer Object (DTO). For example, for a Decision service, the response might resemble:

> **Note**: The "Content-Type" header must be set to "application/json" for the response to the Create URL call.

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}",
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
  }
}
```

This tells Eloqua to send the contacts' Contact ID and email address when the service is executed. For detailed instructions, see the documentation specific to each service type.

3. The marketer can then configure the instance using your service's UI by clicking **Edit** on the service instance. Eloqua opens a browser window to the configure URL for the service. This call to the configure URL comes directly from the marketing user's browser rather than from the Eloqua API. If you need to restrict access, you can do so here by requiring the user be on a specific VPN, for example.

4. If the Marketer has made changes, your API should call back to Eloqua using a `PUT` request with an updated DTO specifying the new record definition. For example, if the Marketer created an AppCloud Decision step that required `field1`, `field2`, and `field3`, the `PUT` request would resemble:

```
PUT https://secure.eloqua.com/api/cloud/1.0/decisions/instance/fddc932a-f27d-40c3-a126-82299b9512c5

{
  "recordDefinition":
  {
    "ContactID" : "{{Contact.Id}}",
    "EmailAddress" : "{{Contact.Field(C_EmailAddress)}}",
    "field1" : "{{Contact.Field(C_field1)}}",
    "field2" : "{{Contact.Field(C_field2)}}",
    "field3" : "{{Contact.Field(C_field3)}}"
  }
}
```

Execution

1. When the service is activated, either by contacts flowing through a campaign or a visit to a landing page, this batch of contacts is tagged with an execution ID that tracks its progress.
2. This step varies depending on the service type, but in general, the Eloqua internal API calls out to the service's notification URL, and transmits a the information you specified during the configuration phase. In the above example, then, the `recordDefinition` includes the contact's ID, email address, field1, field2 and field3. The call to the notification URL would then resemble:

```
POST https://example.com/awesomeapp/decide/notify?instance=fddc932a-f27d-40c3-a126-82299b9512c5&asset=456

{
  "offset" : 0,
  "limit" : 1000,
```

```
 "totalResults" : 2,
 "count" : 2,
 "hasMore" : false,
 "items" :
 [
   {
     "ContactID" : "1",
     "EmailAddress" : "fred@example.com",
     "field1" : "stuff",
     "field2" : "things",
     "field3" : "et cetera"
   },
   {
     "ContactID" : "2",
     "EmailAddress" : "sylvie@example.com",
     "field1" : "more stuff",
     "field2" : "other things",
     "field3" : "and so on"
   }
 ]
}
```

This follows the format defined during the create URL and configure URL calls during the instantiation phase.

3. If Eloqua calls out to your app and doesn't receive a response within 100 seconds, the contacts in a step will be marked as "Errored". If Eloqua receives a response status code that is between 300 and 599, Eloqua will retry the notify call over approximately an eight-hour period of time with a backoff strategy of the time between calls doubling after each call. After this eight-hour period, the contacts in a step will be marked as "Errored". If the

marketer has configured a default path for the contacts to flow through, then the contacts will flow into the next step.

4. If you specified maximum record size of 0 when you configured your app, or the record definition sent during the instantiation phase contains no fields, the DTO contains an empty JSON object. Otherwise, the call sends a DTO containing contact data, with a maximum limit to the number of records as specified in the service configuration. If there are more records than the maximum limit, Eloqua will keep sending requests until all the data is transmitted.
5. You can then choose to return the same response for all the contacts in the batch, or tailor your responses to each contact.

   To individualize your responses for each contact, the desired HTTP response status code is a 204, followed up with an import call using the bulk API. You must specify sync actions during the import. The appropriate sync actions vary according to the service type. Refer to the bulk API documentation and the service-specific documentation for more information
6. Eloqua receives the import. For service instances associated with a campaign, those contacts flow through to the next stage.

Notification URL setup

Many AppCloud services require that you configure a Notification URL when registering the service. The Notification URL is essentially a webhook: it is the URL that Eloqua should call out to when something happens.

For example, when an Eloqua contact visits a landing page containing an AppCloud content service instance, Eloqua calls out to the service provider's Notification URL to request the HTML content needed to populate the landing page. Similarly, when contacts flow into an AppCloud decision or AppCloud action step during a campaign, Eloqua calls out to the notification URL to notify the app that contacts have reached that step.

The notification URL is a templated URL which supports the common parameters, including `{InstanceId}`, `{InstallId}`, `{AssetId}`, `{AssetName}`, `{UserName}`, `{UserId}`, `{EventType}`, and `{UserCulture}`. The AppCloud content service notification URL also supports `{VisitorId}`, which maps to the ID of a visitor when called from a landing page, to enable delivery of visitor-specific content.

When the notification URL is called for AppCloud actions, decisions, feeders, or content, Eloqua also sends, as parameters, the fields specified by your app during the instantiation phase. For example, AwesomeApp has an AppCloud decision service. Its notification URL is:

`https://example.com/awesomeapp/decide/notify?instance={InstanceId}&asset={AssetId}`

During instantiation, ExampleApp specified that it requires the contact ID, email address, field1, field2, and field3 in the *recordDefinition* field of the service instance data transfer object. The *recordDefinition* field was:

```
"recordDefinition": {
  "contactID": "{{Contact.Id}}",
  "email": "{{Contact.Field(C_EmailAddress)}}",
  "field1": "{{Contact.Field(C_Field1)}}",
  "field2": "{{Contact.Field(C_Field2)}}",
  "field3": "{{Contact.Field(C_Field3)}}"
}
```

Suppose that two contacts, fred@example.com and sylvie@example.com, flowed into an AppCloud decision step. Eloqua's call to the notification URL would then resemble:

```
POST https://example.com/awesomeapp/decide/notify?instance=123&asset=456
{
  "offset" : 0,
  "limit" : 1000,
  "totalResults" : 2,
  "count" : 2,
  "hasMore" : false,
  "items" :
  [
    {
     "contactID" : "1",
     "email" : "fred@example.com",
     "field1" : "stuff",
     "field2" : "things",
     "field3" : "et cetera"
    },
    {
     "contactID" : "2",
     "email" : "sylvie@example.com",
     "field1" : "more stuff",
     "field2" : "other things",
     "field3" : "and so on"
    }
   ]
}
```

The appropriate response to the notification URL call varies slightly depending on which service you're developing. You can see the appropriate responses for each service type in the following documents:

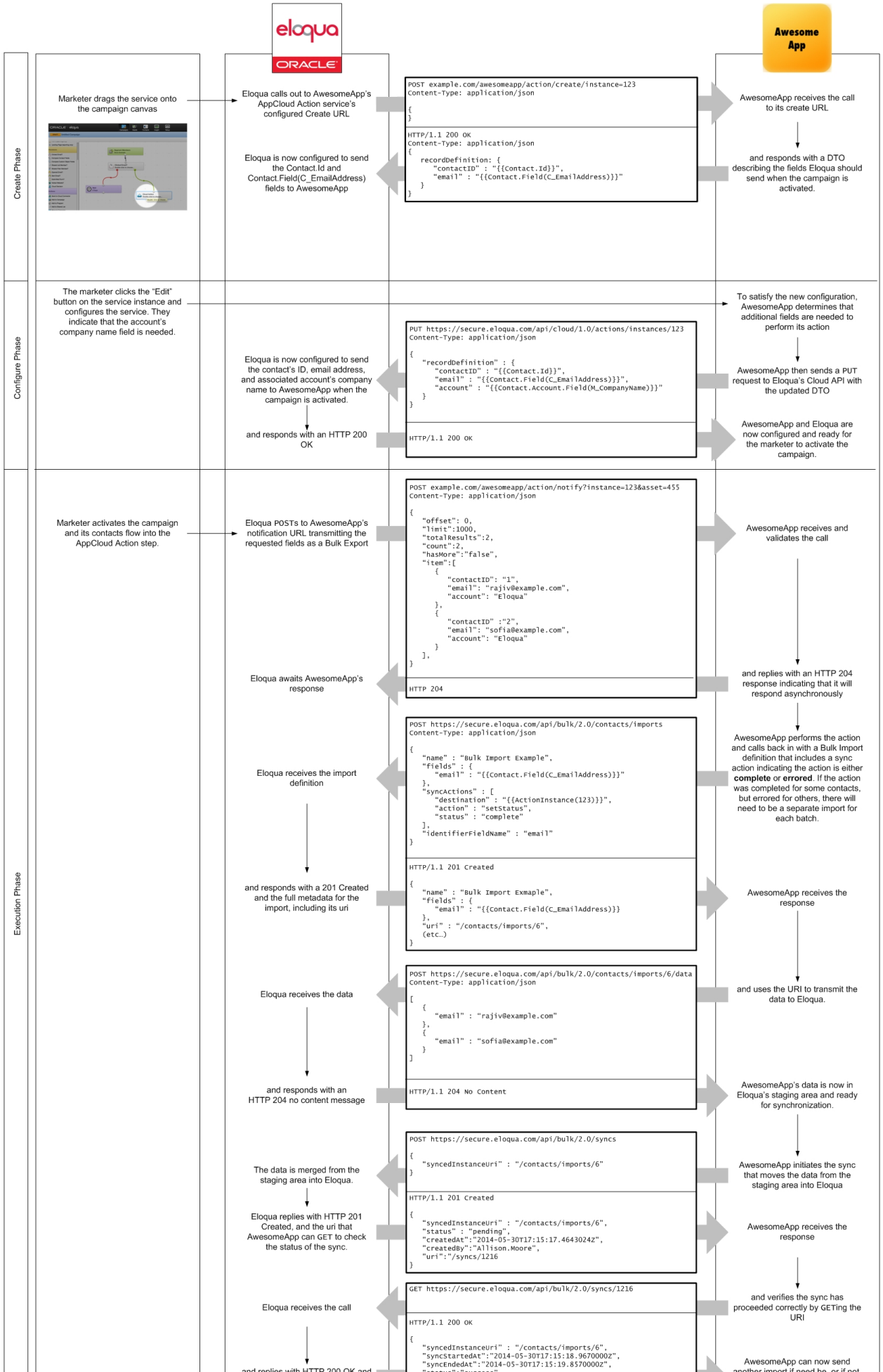
- Develop an AppCloud action service
- Develop an AppCloud decision service
- Develop an AppCloud content service
- Develop an AppCloud feeder service
- Develop an AppCloud firehose service

Develop an Oracle Eloqua app action service

Actions are steps on a campaign or program canvas that are delegated to an external system. This allows you to perform actions that you cannot do with Eloqua. Actions are analogous to the existing cloud connectors.

Actions follow the instantiation-execution model. The following sections describe action-specific details and provide the endpoints needed to develop an Action service. If you are not familiar with the general flow for the instantiation-execution model, you should read that first.

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.


eloqua
ORACLE
Awesome App
Create Phase
Marketer drags the service onto the campaign canvas
Eloqua calls out to AwesomeApp's AppCloud Action service's configured Create URL
POST example.com/awesomeapp/action/create/instance=123
Content-Type: application/json
{
}
HTTP/1.1 200 OK
Content-Type: application/json
{
recordDefinition: {
"contactID" : "{{Contact.Id}}",
"email" : "{{Contact.Field(C_EmailAddress)}}"
}
}
AwesomeApp receives the call to its create URL
and responds with a DTO describing the fields Eloqua should send when the campaign is activated.
Eloqua is now configured to send the Contact.Id and Contact.Field(C_EmailAddress) fields to AwesomeApp
Configure Phase
The marketer clicks the "Edit" button on the service instance and configures the service. They indicate that the account's company name field is needed.
To satisfy the new configuration, AwesomeApp determines that additional fields are needed to perform its action
PUT https://secure.eloqua.com/api/cloud/1.0/actions/instances/123
Content-Type: application/json
{
"recordDefinition" : {
"contactID" : "{{Contact.Id}}",
"email" : "{{Contact.Field(C_EmailAddress)}}",
"account" : "{{Contact.Account.Field(M_CompanyName)}}"
}
}
AwesomeApp then sends a PUT request to Eloqua's Cloud API with the updated DTO
Eloqua is now configured to send the contact's ID, email address, and associated account's company name to AwesomeApp when the campaign is activated.
and responds with an HTTP 200 OK
HTTP/1.1 200 OK
AwesomeApp and Eloqua are now configured and ready for the marketer to activate the campaign.
Execution Phase
Marketer activates the campaign and its contacts flow into the AppCloud Action step.
Eloqua POSTs to AwesomeApp's notification URL transmitting the requested fields as a Bulk Export
POST example.com/awesomeapp/action/notify?instance=123&asset=455
Content-Type: application/json
{
"offset": 0,
"limit":1000,
"totalResults":2,
"count":2,
"hasMore":"false",
"item":[
{
"contactID": "1",
"email": "rajiv@example.com",
"account": "Eloqua"
},
{
"contactID" :"2",
"email": "sofia@example.com",
"account": "Eloqua"
}
],
}
AwesomeApp receives and validates the call
Eloqua awaits AwesomeApp's response
HTTP 204
and replies with an HTTP 204 response indicating that it will respond asynchronously
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports
Content-Type: application/json
{
"name" : "Bulk Import Example",
"fields" : {
"email" : "{{Contact.Field(C_EmailAddress)}}"
},
"syncActions" : [
"destination" : "{{ActionInstance(123)}}",
"action" : "setStatus",
"status" : "complete"
],
"identifierFieldName" : "email"
}
AwesomeApp performs the action and calls back in with a Bulk Import definition that includes a sync action indicating the action is either **complete** or **errored**. If the action was completed for some contacts, but errored for others, there will need to be a separate import for each batch.
Eloqua receives the import definition
and responds with a 201 Created and the full metadata for the import, including its uri
HTTP/1.1 201 Created
{
"name" : "Bulk Import Exmaple",
"fields" : {
"email" : "{{Contact.Field(C_EmailAddress)}}
},
"uri" : "/contacts/imports/6",
(etc…)
}
AwesomeApp receives the response
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports/6/data
Content-Type: application/json
[
{
"email" : "rajiv@example.com"
},
{
"email" : "sofia@example.com"
}
]
Eloqua receives the data
and responds with an HTTP 204 no content message
HTTP/1.1 204 No Content
and uses the URI to transmit the data to Eloqua.
AwesomeApp's data is now in Eloqua's staging area and ready for synchronization.
POST https://secure.eloqua.com/api/bulk/2.0/syncs
{
"syncedInstanceUri" : "/contacts/imports/6"
}
The data is merged from the staging area into Eloqua.
Eloqua replies with HTTP 201 Created, and the uri that AwesomeApp can GET to check the status of the sync.
HTTP/1.1 201 Created
{
"syncedInstanceUri" : "/contacts/imports/6",
"status" : "pending",
"createdAt":"2014-05-30T17:15:17.4643024Z",
"createdBy":"Allison.Moore",
"uri":"/syncs/1216
}
AwesomeApp initiates the sync that moves the data from the staging area into Eloqua
AwesomeApp receives the response
GET https://secure.eloqua.com/api/bulk/2.0/syncs/1216
Eloqua receives the call
and replies with HTTP 200 OK and
HTTP/1.1 200 OK
{
"syncedInstanceUri" : "/contacts/imports/6",
"syncStartedAt":"2014-05-30T17:15:18.9670000Z",
"syncEndedAt":"2014-05-30T17:15:19.8570000Z",
and verifies the sync has proceeded correctly by GETing the URI
AwesomeApp can now send

Create URL

Eloqua's call to the **CreateURL**:

```
POST https://example.com/awesomeapp/act/create?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339

{}
```

AwesomeApp's response is a DTO describing the fields Eloqua should transmit when the service in executed. You can include a maximum of 249 fields in your record definition.

> **Note**: The "Content-Type" header must be set to "application/json" for the response to the Create URL call.

Example response for contacts:

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}",
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
  },
  "requiresConfiguration": true
}
```

Example response for custom objects:

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "Email": "{{CustomObject[1].Field[4]}}"
  },
  "requiresConfiguration": true
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

❑ **Warning**: All field names and values in Eloqua data transfer objects are case sensitive. Be sure to reproduce them exactly. For example: `{{Contact.id}}` would fail if the correct form is `{{Contact.Id}}`.

Configure URL

When a user clicks the edit button on the service instance and makes changes through the **Configure URL** that require an updated recordDefinition data transfer object (DTO), AwesomeApp must call out to Eloqua's cloud API `PUT /actions/instances/{id}` endpoint with that updated DTO:

Example request for contacts:

```
PUT https://secure.p03.eloqua.com/api/cloud/1.0/actions/instances/f82d50cd-86a9-4fca-b37e-4ec9a98b0339

{
   "recordDefinition":
     {
        "ContactID" : "{{Contact.Id}}",
        "EmailAddress" : "{{Contact.Field(C_EmailAddress)}}",
        "field1" : "{{Contact.Field(C_field1)}}",
        "field2" : "{{Contact.Field(C_field2)}}",
        "field3" : "{{Contact.Field(C_field3)}}"
     },
   "requiresConfiguration": false
}
```

Example request for custom objects:

```
PUT https://secure.p03.eloqua.com/api/cloud/1.0/actions/instances/f82d50cd-86a9-4fca-b37e-4ec9a98b0339

{
   "recordDefinition": {
        "Email": "{{CustomObject[1].Field[4]}}"
   },
  "requiresConfiguration": false
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

Use the Configure URL response to set `requiresConfiguration` to `false` when your app's configuration acceptance criteria have been met.

Note: `X-Frame-Options` must not be set to `DENY` for any configure page.

Warning: If the campaign or program is not in draft mode, attempting to set `requiresConfiguration` to `true` will result in an error.

Notification URL

Eloqua calls the **Notification URL** when contacts or custom objects arrive in the action step. This call transmits the requested fields in the `items` parameter:

```
POST https://example.com/awesomeapp/act/notify?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339&asset=456

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 2,
   "count" : 2,
   "hasMore" : false,
   "items" :
   [
     {
       "ContactID" : "1",
       "EmailAddress" : "fred@example.com",
       "field1" : "stuff",
       "field2" : "things",
       "field3" : "et cetera"
     },
     {
       "ContactID" : "2",
```

```
      "EmailAddress" : "john@example.com",
      "field1" : "more stuff",
      "field2" : "other things",
      "field3" : "and so on"
    }
  ]
}
```

Your app must respond to this call, otherwise Eloqua will think the call has timed out.

Note: The amount of records in the `items` parameter is dependent on the **Records per Notification** option during service registration. If you prefer to retrieve records using a separate export, see Retrieving app records using the bulk API for instructions.

Important: If AwesomeApp's Action service is configured to automatically set a contact or custom object record's status to complete, contacts and custom object records will move to the next step in the canvas after a successful response from the app to the notification URL call. Each batch, delivered through a call to the notification URL, of contacts or custom object records in a step will only remain in the workflow for a maximum of 90 days. Attempts to operate on them (set the status to complete or errored) after 90 days will fail, resulting in the records being stuck in the step, and requiring manual intervention to move them along in the flow. Each batch of records sent through a call to the notification

URL has an ExecutionId, available as a template parameter, that identifies a unique batch.

If the action service is not configured to automatically set the status to complete, AwesomeApp should return a 204 response. This indicates that the response will be asynchronous. AwesomeApp should then create an import to Eloqua using the Bulk API where AwesomeApp specifies a sync action: either complete or errored.

Bulk API contact import

1. Create the bulk import definition, setting the status to complete to import data.

   If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a contact sync action definition.

   When importing, the destination field refers to the action service's instance. In this example, the instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`:

   ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

   Create a contact import definition where the sync action sets the record's status, where the instance ID is `f82d50cd-86a9-4fca-b37e4ec9a98b0339` and execution ID is `12345`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports

{
  "name" : "AwesomeApp Action Response Bulk Import",
```

```
  "updateRule" : "always",
  "fields" : {
    "emailAddress" : "{{Contact.Field(C_EmailAddress)}}"
  },
  "syncActions" : [
    {
      "destination" : "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
      "action" : "setStatus",
      "status" : "complete"
    }
  ],
  "identifierFieldName" : "emailAddress"
}
```

Eloqua's response will be a 201 created response that includes a `uri` parameter, which you can use to identify the import:

```
{
    "name" : "AwesomeApp Action Response Bulk Import",
    "updateRule" : "always",
    "fields" : {
      "emailAddress" : "{{Contact.Field(C_EmailAddress)}}"
    },
    "identifierFieldName" : "emailAddress",
    "syncActions" : [
      {
        "destination" : "{{ActionInstance
```

```
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
          "action" : "setStatus",
          "status" : "complete"
        }
      ],
      "isSyncTriggeredOnImport" : false,
      "isUpdatingMultipleMatchedRecords" : false,
      "uri" : "/contacts/imports/6",
      "createdBy" : "DocsExample",
      "createdAt" : "2014-03-06T13:59:00.6600046Z",
      "updatedBy" : "DocsExample",
      "updatedAt" : "2014-03-06T13:59:00.6600046Z"
  }
```

2. Send Eloqua the import data using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/6/data
```

```
[
    {
      "emailAddress" : "fred@example.com"
    },
    {
      "emailAddress" : "sylvie@example.com"
    }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/contacts/imports/6"
}
```

Eloqua's response:

```
201 Created


{
    "syncedInstanceURI" : "/contacts/imports/6",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

4. You can then use the sync's URI (`/syncs/6`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/6
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
     "syncedInstanceURI" : "/contacts/imports/6",
     "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
     "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
```

```
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

> ✍ **Learn more**: Bulk API imports.

Bulk API custom object import

1. Create the bulk import definition, setting the status to complete to import data.

   If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a custom object sync action definition.

   When importing, the destination field refers to the action service's instance - in this example, the instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`:

   > ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

   Create a custom object import definition where the sync action sets the record's status, where the instance ID is `f82d50cd-86a9-4fca-b37e4ec9a98b0339` and execution ID is `12345`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/9/imports

{
  "name" : "AwesomeApp Action Response Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "email" : "{{CustomObject[9].Field[58]}}"
  },
  "syncActions" : [
    {
      "destination" : "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
      "action" : "setStatus",
      "status" : "complete"
    }
  ],
  "identifierFieldName" : "email"
}
```

Eloqua's response will be a 201 created response that includes a `uri` parameter, which you can use to identify the import:

```
  HTTP/1.1 201 Created

  {
      "name" : "AwesomeApp Action Response Bulk Import",
      "updateRule" : "always",
      "fields" : {
        "email" : "{{CustomObject[9].Field[58]}}"
      },
```

```
    "identifierFieldName" : "email",
    "syncActions" : [
      {
        "destination" : "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
        "action" : "setStatus",
        "status" : "complete"
      }
    ],
    "isSyncTriggeredOnImport" : false,
    "isUpdatingMultipleMatchedRecords" : false,
    "uri" : "/customObjects/imports/9",
    "createdBy" : "DocsExample",
    "createdAt" : "2014-03-06T13:59:00.6600046Z",
    "updatedBy" : "DocsExample",
    "updatedAt" : "2014-03-06T13:59:00.6600046Z"
}
```

2. Send Eloqua the data for import as a using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/imports/9/data
```

```
[
  {
    "email" : "fred@example.com"
  },
  {
    "email" : "sylvie@example.com"
  }
```

```
]
```

3. Synchronize the data for import:

   AwesomeApp request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/customObjects/imports/9"
}
```

   Eloqua's response:

```
201 Created


{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

4. You can then use the sync's uri (`/syncs/9`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/9
```

   When the sync is complete, Eloqua's response will resemble:

```
200 OK

{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

✍ **Learn more**: Bulk API imports.

Delete URL

The delete URL is a templated URL pointing to an endpoint for deleting an instance of your service.

The delete URL uses an HTTP DELETE request and there is no content sent in the request body. All common URL template parameters are available (the same as with a create URL). On success, this endpoint should return a 200-level response.

An example delete URL would look something like:

```
https://www.someurl.com/delete/{appId}/{installId}/{instanceId}/{userName}/{siteName}/{siteId}
```

Note: Delete calls are not sent in real time, but are done in a batch once daily.

Learn more

Develop an Oracle Eloqua app decision service

Decisions are steps on a campaign or program canvas that directly control the path a contact or custom object takes as they flow through a canvas. Decision steps delegate the decision making to an external system, which responds with a yes or no for each object flowing through the canvas, determining which path the contact or custom object should take.

Decisions follow the instantiation-execution model. The following sections describe decision-specific details and provide the endpoints needed to develop a decision service. If you are not familiar with the general flow for the instantiation-execution model, you should read that first.

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

Create Phase

Marketer drags the service onto the campaign canvas

Eloqua calls out to AwesomeApp's AppCloud Decision service's configured Create URL

```
POST example.com/awesomeapp/decision/create/instance=123
Content-Type: application/json

{
}
```

AwesomeApp receives the call to its create URL

and responds with a DTO describing the fields Eloqua should send when the campaign is activated.

```
HTTP/1.1 200 OK
Content-Type: application/json
{
   recordDefinition: {
      "contactID" : "{{Contact.Id}}",
      "email" : "{{Contact.Field(C_EmailAddress)}}"
   }
}
```

Eloqua is now configured to send the Contact.Id and Contact.Field(C_EmailAddress) fields to AwesomeApp

Configure Phase

The marketer clicks the "Edit" button on the service instance and configures the service. They indicate that the account's company name field is needed.

To satisfy the new configuration, AwesomeApp determines that additional fields are needed to perform its decision

AwesomeApp then sends a PUT request to Eloqua's Cloud API with the updated DTO

```
PUT https://secure.eloqua.com/api/cloud/1.0/decisions/instances/123
Content-Type: application/json

{
   "recordDefinition" : {
      "contactID" : "{{Contact.Id}}",
      "email" : "{{Contact.Field(C_EmailAddress)}}",
      "account" : "{{Contact.Account.Field(M_CompanyName)}}"
   }
}
```

Eloqua is now configured to send the contact's ID, email address, and associated account's company name to AwesomeApp when the campaign is activated.

and responds with an HTTP 200 OK

```
HTTP/1.1 200 OK
```

AwesomeApp and Eloqua are now configured and ready for the marketer to activate the campaign.

Execution Phase

Marketer activates the campaign and its contacts flow into the AppCloud Decision step.

Eloqua POSTs to AwesomeApp's notification URL transmitting the requested fields as a Bulk Export

```
POST example.com/awesomeapp/decision/notify?instance=123&asset=455
Content-Type: application/json

{
   "offset": 0,
   "limit":1000,
   "totalResults":2,
   "count":2,
   "hasMore":"false",
   "item":[
      {
         "contactID": "1",
         "email": "rajiv@example.com",
         "account": "Eloqua"
      },
      {
         "contactID" :"2",
         "email": "sofia@example.com",
         "account": "Eloqua"
      }
   ],
}
```

AwesomeApp receives and validates the call

and replies with an HTTP 204 response indicating that it will respond asynchronously

```
HTTP 204
```

Eloqua awaits AwesomeApp's response

AwesomeApp determines the correct decision response and calls back in with a Bulk Import definition that includes a sync action indicating the decision is either **yes**, **no**, or **errored**. There will need to be a separate import for each batch – one for the 'yes', one for the 'no', and one for any 'errored' records..

```
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports
Content-Type: application/json

{
   "name" : "Bulk Import Example",
   "fields" : {
      "email" : "{{Contact.Field(C_EmailAddress)}}"
   },
   "syncActions" : [
      "destination" : "{{DecisionInstance(123)}}",
      "action" : "setStatus",
      "status" : "yes"
   ],
   "identifierFieldName" : "email"
}
```

Eloqua receives the import definition

and responds with a 201 Created and the full metadata for the import, including its uri

```
HTTP/1.1 201 Created

{
   "name" : "Bulk Import Exmaple",
   "fields" : {
      "email" : "{{Contact.Field(C_EmailAddress)}}
   },
   "uri" : "/contacts/imports/6",
   (etc…)
}
```

AwesomeApp receives the response

```
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports/6/data
Content-Type: application/json

[
   {
      "email" : "rajiv@example.com"
   },
   {
      "email" : "sofia@example.com"
   }
]
```

Eloqua receives the data

and uses the URI to transmit the data to Eloqua.

```
HTTP/1.1 204 No Content
```

and responds with an HTTP 204 No Content message

AwesomeApp's data is now in Eloqua's staging area and ready for synchronization.

```
POST https://secure.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceUri" : "/contacts/imports/6"
}
```

The data is merged from the staging area into Eloqua.

AwesomeApp initiates the sync that moves the data from the staging area into Eloqua

```
HTTP/1.1 201 Created

{
   "syncedInstanceUri" : "/contacts/imports/6",
   "status" : "pending",
   "createdAt":"2014-05-30T17:15:17.4643024Z",
   "createdBy":"Docs.Example",
   "uri":"/syncs/1216
}
```

Eloqua replies with HTTP 201 Created, and the uri that AwesomeApp can GET to check the status of the sync.

AwesomeApp receives the response

```
GET https://secure.eloqua.com/api/bulk/2.0/syncs/1216
```

Eloqua receives the call

and verifies the sync has proceeded correctly by GETing the URI

```
HTTP/1.1 200 OK

{
   "syncedInstanceUri" : "/contacts/imports/6",
   "syncStartedAt":"2014-05-30T17:15:18.9670000Z",
   "syncEndedAt":"2014-05-30T17:15:19.8570000Z",
```

and replies with HTTP 200 OK and

AwesomeApp can now send another import if need be, or if not

Create URL

Eloqua's call to the **CreateURL**:

```
POST https://example.com/awesomeapp/decide/create?instance=9347bfe1-9c72-409c-a5cd-402ff74f0caa

{}
```

AwesomeApp's response is a DTO describing the fields Eloqua should transmit when the service in executed. You can include a maximum of 249 fields in your record definition.

> **Note**: The "Content-Type" header must be set to "application/json" for the response to the Create URL call.

Example response for contacts:

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}",
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
  },
  "requiresConfiguration": true
}
```

Example response for custom objects:

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "Email": "{{CustomObject[1].Field[4]}}"
  },
  "requiresConfiguration": true
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

❑ **Warning**: All field names and values in Eloqua data transfer objects are case sensitive. Be sure to reproduce them exactly. For example: `{{Contact.id}}` would fail if the correct form is `{{Contact.Id}}`.

Configure URL

When a user clicks the edit button on the service instance and makes changes through the **Configure URL** that require an updated recordDefinition DTO, AwesomeApp must call out to Eloqua's cloud API `PUT /decisions/instances/{id}` endpoint with that updated DTO:

Example request for contacts:

```
PUT https://secure.p03.eloqua.com/api/cloud/1.0/decisions/instances/9347bfe1-9c72-409c-a5cd-402ff74f0caa

{
   "recordDefinition":
     {
        "ContactID" : "{{Contact.Id}}",
        "EmailAddress" : "{{Contact.Field(C_EmailAddress)}}",
        "field1" : "{{Contact.Field(C_field1)}}",
        "field2" : "{{Contact.Field(C_field2)}}",
        "field3" : "{{Contact.Field(C_field3)}}"
     },
   "requiresConfiguration": false
}
```

Example request for custom objects:

```
PUT https://secure.p03.eloqua.com/api/cloud/1.0/decisions/instances/9347bfe1-9c72-409c-a5cd-402ff74f0caa

{
   "recordDefinition": {
        "Email": "{{CustomObject[1].Field[4]}}"
   },
  "requiresConfiguration": false
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

Use the Configure URL response to set `requiresConfiguration` to `false` when your app's configuration acceptance criteria have been met.

Note: `X-Frame-Options` must not be set to `DENY` for any configure page.

Warning: If the campaign or program is not in draft mode, attempting to set `requiresConfiguration` to `true` will result in an error.

Notification URL

Eloqua's call to the **Notification URL** transmits the requested fields in the items parameter:

```
POST https://example.com/awesomeapp/decide/notify?instance=9347bfe1-9c72-409c-a5cd-402ff74f0caa&asset=456

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 2,
   "count" : 2,
   "hasMore" : false,
   "items" :
   [
     {
       "ContactID" : "1",
       "EmailAddress" : "fred@example.com",
       "field1" : "stuff",
       "field2" : "things",
       "field3" : "et cetera"
     },
     {
       "ContactID" : "2",
       "EmailAddress" : "john@example.com",
```

```
      "field1" : "more stuff",
      "field2" : "other things",
      "field3" : "and so on"
    }
  ]
}
```

Note: The amount of records in the `items` parameter is dependent on the **Records per Notification** option during service registration. If you prefer to retrieve records using a separate export, see Retrieving app records using the bulk API for instructions.

Important: Each batch, delivered through a call to the notification URL, of contacts or custom object records in a step will only remain in the workflow for a maximum of 90 days. Attempts to operate on them (set the status to complete or errored) after 90 days will fail, resulting in the records being stuck in the step, and requiring manual intervention to move them along in the flow. Each batch of records sent through a call to the notification URL has an ExecutionId, available as a template parameter, that identifies a unique batch.

AwesomeApp responds with a 204 response, indicating that the response will be asynchronous, followed by an **import** to Eloqua's Bulk API, where AwesomeApp specifies a sync action. For Decisions, the sync action is either "**yes**", "**no**", or "**errored**". AwesomeApp will need to create multiple imports for the different statuses,

with contacts or custom objects that should follow the "no" path in one import and then contacts and custom objects that should follow the "yes" path in another. Note that the maximum import size per batch is 5,000: if you have more than 5,000 contacts or custom objects, break your data up into multiple imports.

Bulk API contact import

1. Create the bulk import definition, setting the status to **yes** to import data.

   If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a contact sync action definition.

   When importing, the "destination" field refers to the AppCloud Decision service's instance - in this example, the instance ID is `9347bfe1-9c72-409c-a5cd-402ff74f0caa`:

   > ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

   Create a contact import definition where the sync action sets the record's status, where the instance ID is `9347bfe1-9c72-409c-a5cd-402ff74f0caa` and execution ID is `12345`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports

{
  "name" : "AwesomeApp Decision Response Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "emailAddress" : "{{Contact.Field(C_EmailAddress)}}"
  }
  "syncActions" : [
```

```
    {
      "destination" : "{{DecisionInstance
(9347bfe19c72409ca5cd402ff74f0caa).Execution[12345]}}",
      "action" : "setStatus",
      "status" : "yes"
    }
  ],
  "identifierFieldName" : "emailAddress"
}
```

Eloqua's response will be a 201 created response that includes a `uri` parameter, which you can use to identify the import:

```
{
  "name": "AwesomeApp Decision Response Bulk Import",
  "updateRule": "always",
  "fields": {
    "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
  },
  "identifierFieldName": "emailAddress",
  "syncActions": [
    {
      "destination": "{{DecisionInstance
(9347bfe19c72409ca5cd402ff74f0caa).Execution[12345]}}",
      "action": "setStatus",
      "status": "yes"
    }
  ],
```

```
  "isSyncTriggeredOnImport": false,
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/contacts/imports/6",
  "createdBy": "DocsExample",
  "createdAt": "2014-03-06T13:59:00.6600046Z",
  "updatedBy": "DocsExample",
  "updatedAt": "2014-03-06T13:59:00.6600046Z"
}
```

2. Send Eloqua the import data using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/6/data
```

```
[
   {
     "emailAddress" : "fred@example.com"
   },
   {
     "emailAddress" : "sylvie@example.com"
   }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/contacts/imports/6"
}
```

Eloqua's response:

```
201 Created


{
    "syncedInstanceURI" : "/contacts/imports/6",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

4. You can then use the sync's URI (`/syncs/6`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/6
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
    "syncedInstanceURI" : "/contacts/imports/6",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

Bulk API custom object import

1. Create the bulk import definition, setting the status to **yes** to import data.

   If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a custom object sync action definition.

   When importing, the destination field refers to the decision service's instance - in this example, the instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`:

   > ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

   Create a custom object import definition where the sync action sets the record's status, where the instance ID is `f82d50cd-86a9-4fca-b37e4ec9a98b0339` and execution ID is `12345`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/9/imports

{
  "name" : "AwesomeApp Decision Response Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "email" : "{{CustomObject[9].Field[58]}}"
  },
  "syncActions" : [
    {
       "destination" : "{{DecisionInstance
```

```
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
        "action" : "setStatus",
        "status" : "yes"
    }
  ],
  "identifierFieldName" : "email"
}
```

Eloqua's response will be a 201 created response that includes a `uri` parameter, which you can use to identify the import:

```
HTTP/1.1 201 Created

{
    "name" : "AwesomeApp Decision Response Bulk Import",
    "updateRule" : "always",
    "fields" : {
      "email" : "{{CustomObject[9].Field[58]}}"
    },
    "identifierFieldName" : "email",
    "syncActions" : [
      {
        "destination" : "{{DecisionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
        "action" : "setStatus",
        "status" : "yes"
      }
    ],
    "isSyncTriggeredOnImport" : false,
```

```
    "isUpdatingMultipleMatchedRecords" : false,
    "uri" : "/customObjects/imports/9",
    "createdBy" : "DocsExample",
    "createdAt" : "2014-03-06T13:59:00.6600046Z",
    "updatedBy" : "DocsExample",
    "updatedAt" : "2014-03-06T13:59:00.6600046Z"
  }
```

2. Send Eloqua the data for import as a using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/imports/9/data
```

```
[
   {
     "email" : "fred@example.com"
   },
   {
     "email" : "sylvie@example.com"
   }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/customObjects/imports/9"
}
```

Eloqua's response:

```
201 Created


{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

4. You can then use the sync's URI (`/syncs/9`) to check the status of the sync:

```
GET https://secure.p03.eloqua/api/bulk/2.0/sync/9
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

> **Learn more**: Bulk API imports.

Delete URL

The delete URL is a templated URL pointing to an endpoint for deleting an instance of your service.

The delete URL uses an HTTP DELETE request and there is no content sent in the request body. All common URL template parameters are available (the same as with a create URL). On success, this endpoint should return a 200-level response.

An example delete URL would look something like:

```
https://www.someurl.com/delete/{appId}/{installId}/{instanceId}/{userName}/
{siteName}/{siteId}
```

> **Note**: Delete calls are not sent in real time, but are done in a batch once daily.

Develop an Oracle Eloqua app content service

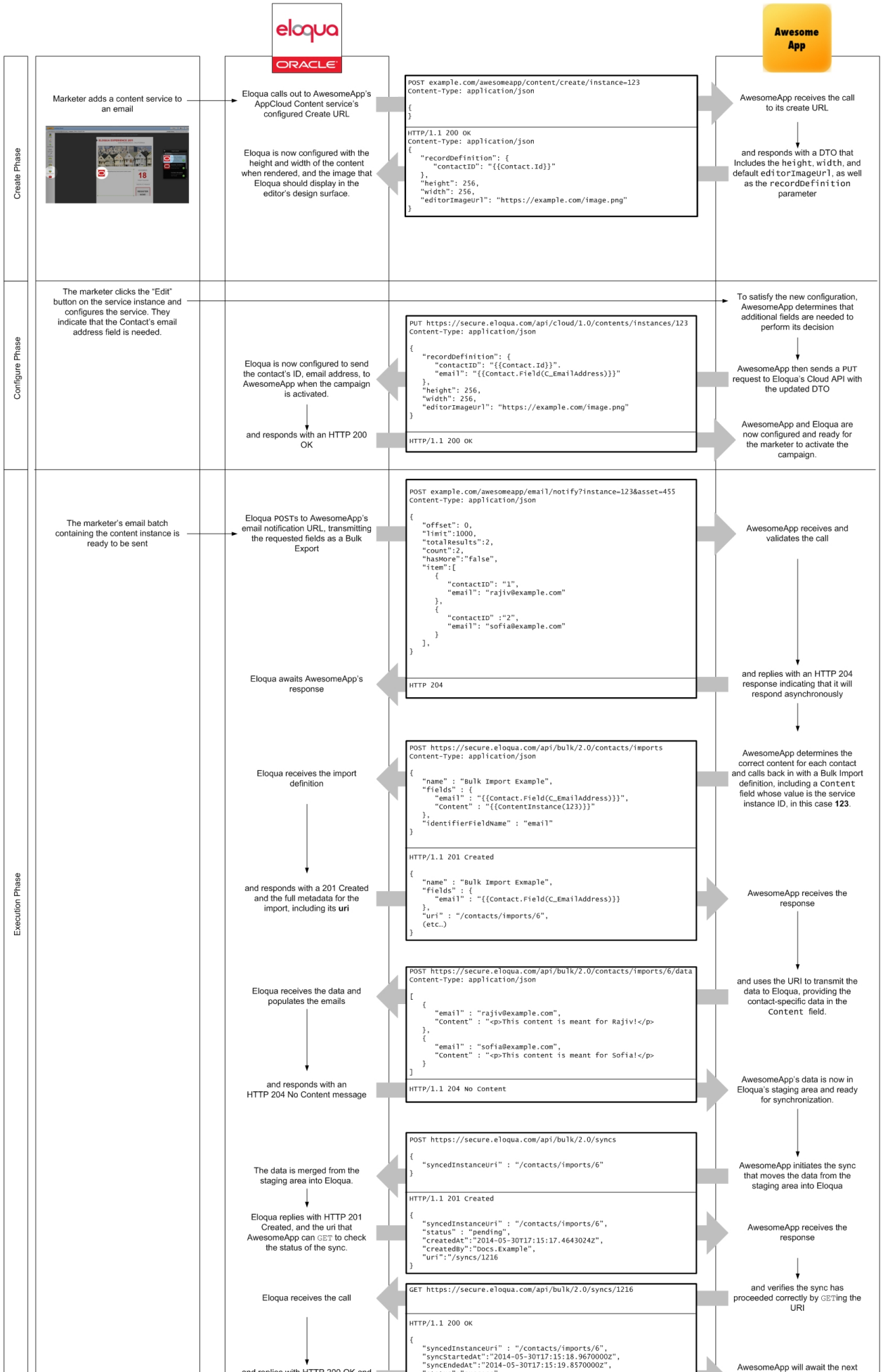
Content allows marketers to source pieces of content in Eloqua emails and landing pages, from an external source. This is the new and improved version of Eloqua's cloud components framework, and it includes many improvements such as asynchronous bulk processing (for emails), the ability to fully test the content service within Eloqua, and design-time interaction with the email and landing page editors.

Content follows the instantiation-execution model.The following sections describe content-specific details and provide the endpoints needed to develop a content service. If you are not familiar with the general flow for the instantiation-execution model, you should read that first.

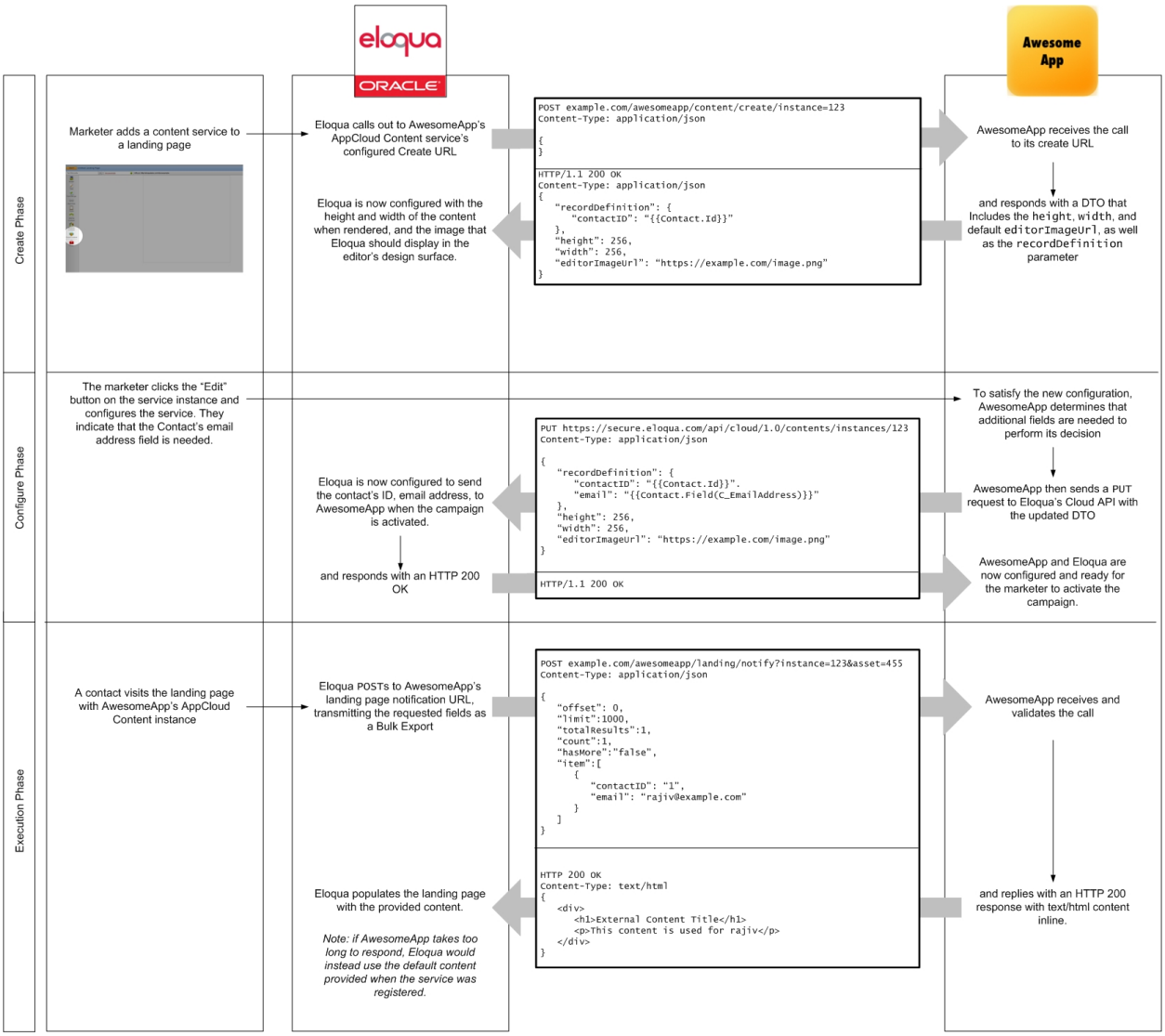
Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

Content service flow diagrams

For emails


eloqua
ORACLE
Awesome App
Create Phase
Marketer adds a content service to an email
Eloqua calls out to AwesomeApp's AppCloud Content service's configured Create URL
POST example.com/awesomeapp/content/create/instance=123
Content-Type: application/json
{
}
HTTP/1.1 200 OK
Content-Type: application/json
{
"recordDefinition": {
"contactID": "{{Contact.Id}}"
},
"height": 256,
"width": 256,
"editorImageUrl": "https://example.com/image.png"
}
AwesomeApp receives the call to its create URL
and responds with a DTO that Includes the height, width, and default editorImageUrl, as well as the recordDefinition parameter
Eloqua is now configured with the height and width of the content when rendered, and the image that Eloqua should display in the editor's design surface.
Configure Phase
The marketer clicks the "Edit" button on the service instance and configures the service. They indicate that the Contact's email address field is needed.
To satisfy the new configuration, AwesomeApp determines that additional fields are needed to perform its decision
PUT https://secure.eloqua.com/api/cloud/1.0/contents/instances/123
Content-Type: application/json
{
"recordDefinition": {
"contactID": "{{Contact.Id}}".
"email": "{{Contact.Field(C_EmailAddress)}}"
},
"height": 256,
"width": 256,
"editorImageUrl": "https://example.com/image.png"
}
HTTP/1.1 200 OK
AwesomeApp then sends a PUT request to Eloqua's Cloud API with the updated DTO
Eloqua is now configured to send the contact's ID, email address, to AwesomeApp when the campaign is activated.
and responds with an HTTP 200 OK
AwesomeApp and Eloqua are now configured and ready for the marketer to activate the campaign.
Execution Phase
The marketer's email batch containing the content instance is ready to be sent
Eloqua POSTs to AwesomeApp's email notification URL, transmitting the requested fields as a Bulk Export
POST example.com/awesomeapp/email/notify?instance=123&asset=455
Content-Type: application/json
{
"offset": 0,
"limit":1000,
"totalResults":2,
"count":2,
"hasMore":"false",
"item":[
{
"contactID": "1",
"email": "rajiv@example.com"
},
{
"contactID" :"2",
"email": "sofia@example.com"
}
],
}
HTTP 204
AwesomeApp receives and validates the call
and replies with an HTTP 204 response indicating that it will respond asynchronously
Eloqua awaits AwesomeApp's response
Eloqua receives the import definition
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports
Content-Type: application/json
{
"name" : "Bulk Import Example",
"fields" : {
"email" : "{{Contact.Field(C_EmailAddress)}}",
"Content" : "{{ContentInstance(123)}}"
},
"identifierFieldName" : "email"
}
HTTP/1.1 201 Created
{
"name" : "Bulk Import Exmaple",
"fields" : {
"email" : "{{Contact.Field(C_EmailAddress)}}
},
"uri" : "/contacts/imports/6",
(etc…)
}
AwesomeApp determines the correct content for each contact and calls back in with a Bulk Import definition, including a Content field whose value is the service instance ID, in this case 123.
and responds with a 201 Created and the full metadata for the import, including its uri
AwesomeApp receives the response
Eloqua receives the data and populates the emails
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports/6/data
Content-Type: application/json
[
{
"email" : "rajiv@example.com",
"Content" : " This content is meant for Rajiv!
},
{
"email" : "sofia@example.com",
"Content" : " This content is meant for Sofia!
}
]
HTTP/1.1 204 No Content
and uses the URI to transmit the data to Eloqua, providing the contact-specific data in the Content field.
and responds with an HTTP 204 No Content message
AwesomeApp's data is now in Eloqua's staging area and ready for synchronization.
The data is merged from the staging area into Eloqua.
POST https://secure.eloqua.com/api/bulk/2.0/syncs
{
"syncedInstanceUri" : "/contacts/imports/6"
}
HTTP/1.1 201 Created
{
"syncedInstanceUri" : "/contacts/imports/6",
"status" : "pending",
"createdAt":"2014-05-30T17:15:17.4643024Z",
"createdBy":"Docs.Example",
"uri":"/syncs/1216
}
AwesomeApp initiates the sync that moves the data from the staging area into Eloqua
Eloqua replies with HTTP 201 Created, and the uri that AwesomeApp can GET to check the status of the sync.
AwesomeApp receives the response
Eloqua receives the call
GET https://secure.eloqua.com/api/bulk/2.0/syncs/1216
HTTP/1.1 200 OK
{
"syncedInstanceUri" : "/contacts/imports/6",
"syncStartedAt":"2014-05-30T17:15:18.9670000Z",
"syncEndedAt":"2014-05-30T17:15:19.8570000Z",
and verifies the sync has proceeded correctly by GETing the URI
AwesomeApp will await the next

For landing pages



Create URL

Eloqua's call to the **CreateURL**:

```
POST https://example.com/awesomeapp/content/create?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339

{}
```

For a content service, AwesomeApp's response must include `height`, `width`, and default `editorImageUrl`, as well as the `recordDefinition` parameter. You can include a maximum of 250 fields in your record definition.

The `height` and `width` parameters define the size of the content instance when rendered, while `editorImageUrl` specifies the URL for an image that Eloqua will display in the editor's design surface. `editorImageUrl` is **not** a templated URL.

> **Note**: The "Content-Type" header must be set to "application/json" for the response to the Create URL call.

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}"
  },
  "height": 256,
  "width": 256,
  "editorImageUrl": "https://example.com/image.png",
  "requiresConfiguration": true
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to save an email or landing page asset containing the unconfigured app service instance. Eloqua will display an error message.

❑ **Warning**: All field names and values in Eloqua data transfer objects are case sensitive. Be sure to reproduce them exactly. For example: `{{Contact.id}}` would fail if the correct form is `{{Contact.Id}}`.

Configure URL

When a user clicks the edit button on the service instance and makes changes through the **Configure URL** that require an updated recordDefinition DTO, AwesomeApp must call out to Eloqua's cloud API `PUT /contents/instances/{id}` endpoint with that updated DTO:

```
PUT https://secure.eloqua.com/api/cloud/1.0/contents/instances/f82d50cd-86a9-4fca-b37e-4ec9a98b0339

{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}",
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
  },
  "height": 256,
  "width": 256,
  "editorImageUrl": "https://example.com/image.png",
  "requiresConfiguration": false
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to save an email or landing page asset containing the unconfigured app service instance. Eloqua will display an error message.

Use the configure URL response to set `requiresConfiguration` to `false` when your app's configuration acceptance criteria have been met.

> **Note**: `X-Frame-Options` must not be set to `DENY` for any configure page.

> **Warning**: If the email or landing page asset is not in draft mode, attempting to set `requiresConfiguration` to `true` will result in an error.

Notification URL

Eloqua's call to the **Notification URL** transmits the requested fields in the `items` parameter:

```
POST https://example.com/awesomeapp/content/notify?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339&asset=456

{
   "offset": 0,
   "limit": 1000,
   "totalResults": 2,
   "count": 2,
   "hasMore": false,
   "items":
   [
     {
       "ContactID": "1",
       "EmailAddress": "fred@example.com"
     },
     {
       "ContactID": "2",
       "EmailAddress": "john@example.com"
```

```
    }
  ]
}
```

For content services, AwesomeApp's response will depend on whether the content is for an email or a landing page:

For landing pages, the response is a 200 status code with text/html content inline. If the response takes too long, Eloqua uses the default content for that contact.

For **Email**, AwesomeApp can respond in one of two ways:

- *Inline response*: A 200 status code with text/html content inline, in which case Eloqua uses the same content for each contact (this is the same response type as used for landing pages), **or**
- *Asynchronous response*: Asynchronous response: A 204 status code, indicating that the call was accepted, but there is no content to return directly. The service should process asynchronously and then call the bulk API. Eloqua waits 24 hours for a response. If there is no response from the app after 24 hours the email is not sent and those records are set to error.

> **Important**: If there is no response or an error returned for the Notification Call Eloqua uses the default content for that contact.

Inline response

The inline response for landing pages and email is a 200 status code, followed by the text/html content. For example:

```
{
  <div>
    <h1>External Content</h1>
    <p>This content is used for all recipients for this request.</p>
  </div>
}
```

Asynchronous response

For the asynchronous response, AwesomeApp responds with a 204 status code, indicating that the response will be asynchronous, followed by one or more imports to Eloqua's bulk API, where contact-specific content is updated with the ContactId and the instance Id, mapping the contact to the new html content.

Note: The maximum import size per batch is 5,000. If you have more than 5,000 contacts, break your data up into multiple imports.

1. Create the bulk import definition, including a `Content` parameter whose value is the service instance ID and execution ID. Including the execution ID enables you to differentiate between different uses of the asset, for example, when multiple campaigns use the same email template that contains the service instance:

```
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports
{
  "name": "AwesomeApp Content Response Bulk Import",
  "updateRule": "always",
  "fields": {
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
```

```
      "Content": "{{ContentInstance
  (f82d50cd86a94fcab37e4ec9a98b0339).Execution[21]}}"
    },
    "identifierFieldName": "EmailAddress"
  }
```

Eloqua's response will be a 201 Created response that includes a `uri` parameter, which you can use to identify the import:

```
HTTP/1.1 201 Created
{
  "name" : "AwesomeApp Content Response Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "EmailAddress" : "{{Contact.Field(C_EmailAddress)}}",
    "Content" : "{{ContentInstance(f82d50cd86a94fcab37e4ec9a98b0339).Execution
[21]}}"
  },
  "identifierFieldName" : "EmailAddress",
  "isSyncTriggeredOnImport" : false,
  "isUpdatingMultipleMatchedRecords" : false,
  "uri" : "/contacts/imports/6",
  "createdBy" : "DocsExample",
  "createdAt" : "2014-03-06T13:59:00.6600046Z",
  "updatedBy" : "DocsExample",
  "updatedAt" : "2014-03-06T13:59:00.6600046Z"
}
```

2. Send Eloqua the data for import, using the content parameter with the content the Eloqua should use for each contact:

```
POST https://secure.eloqua.com/api/bulk/2.0/contacts/imports/6/data
[
  {
    "EmailAddress" : "fred@example.com",
    "Content" : "<p>This is the content for Fred</p>"
  },
  {
    "EmailAddress" : "sylvie@example.com",
    "Content" : "<p>This is the content for Sylvie</p>"
  }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/contacts/imports/6"
}
```

   Eloqua's response:

```
201 Created


{
    "syncedInstanceURI" : "/contacts/imports/6",
```

```
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

4. You can then use the sync's URI (`/syncs/6`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/6
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
    "syncedInstanceURI" : "/contacts/imports/6",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

Notes:

- If you want your content service hyperlinks to take advantage of Eloqua's dynamic link tracking features, it will not do so automatically. To enable Eloqua tracking for a link within your content service:
  - If the destination page has an Eloqua tracking script installed you should append `s=<span class=eloquaemail>siteid</span>&elq=<span class=eloquaemail>recipientid</span>` to the URL.
  - If the destination page does not have an Eloqua tracking script installed you should append `elqtrack=true` to the URL.
- The http protocol must be included to use Eloqua tracking parameters. For example:
  - `google.com?elqTrack=true` - Detected as invalid URL. No clickthrough is recorded.
  - `http://google.com?elqTrack=true` - Detected as valid URL. Clickthrough is recorded.

Delete URL

The delete URL is a templated URL pointing to an endpoint for deleting an instance of your service.

The delete URL uses an HTTP DELETE request and there is no content sent in the request body. All common URL template parameters are available (the same as with a create URL). On success, this endpoint should return a 200-level response.

An example delete URL would look something like:

```
https://www.someurl.com/delete/{appId}/{installId}/{instanceId}/{userName}/{siteName}/{siteId}
```

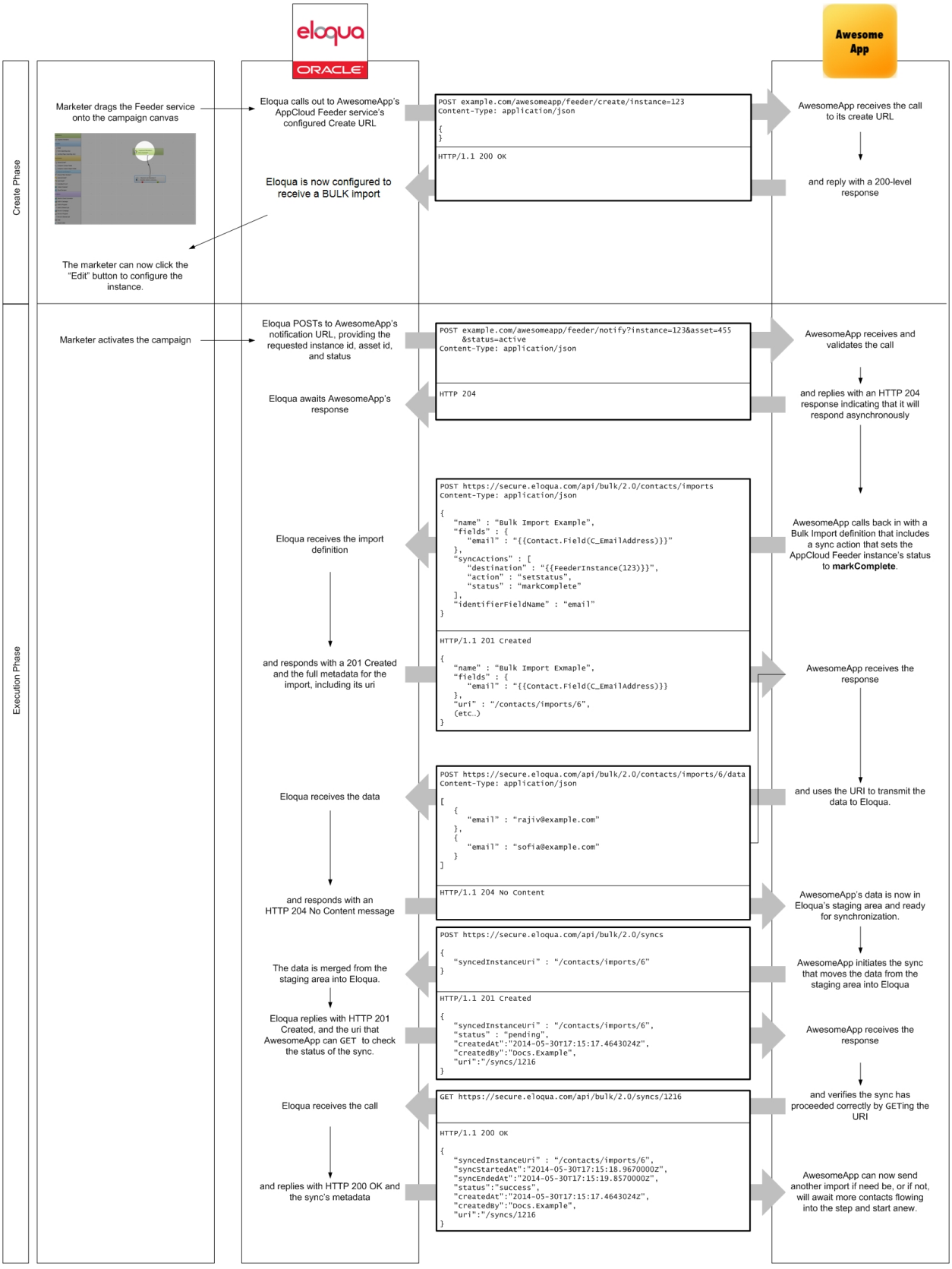
Note: Delete calls are not sent in real time, but are done in a batch once daily.

Develop an Oracle Eloqua app feeder service

Feeders allow external systems to determine and control which contacts or custom objects enter into a campaign or program canvas, and when. Developers can now build apps that utilize data in third-party systems to drive campaign or program membership.

Feeders follow the instantiation-execution model. The following sections describe feeder-specific details and provide the endpoints needed to develop a feeder service. If you are not familiar with the general flow for the instantiation-execution model, you should read that first.

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.



Create URL

Eloqua's call to the **CreateURL**:

```
POST https://example.com/awesomeapp/feeders/create?siteid=123instance=9347bfe1-9c72-409c-a5cd-402ff74f0caa

{}
```

AwesomeApp's response is a 200-level HTTP response:

> **Note**: The "Content-Type" header must be set to "application/json" for the response to the Create URL call.

```
HTTP/1.1 200 application/json; charset=utf-8

{
  "requiresConfiguration": true
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

Configure URL

When a user clicks the edit button on the service instance and makes changes through the **Configure URL**, AwesomeApp must call out to Eloqua's Cloud API `PUT /feeders/instances/{id}` endpoint:

```
PUT https://secure.eloqua.com/api/cloud/1.0/feeders/instances/9347bfe1-9c72-409c-a5cd-402ff74f0caa
```

```
{
  "requiresConfiguration": false
}
```

`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

Use the Configure URL response to set `requiresConfiguration` to `false` when your app's configuration acceptance criteria have been met.

↖**Note**: `X-Frame-Options` must not be set to `DENY` for any configure page.

❑ **Warning**: If the campaign or program is not in draft mode, attempting to set `requiresConfiguration` to `true` will result in an error.

Notification URL

Eloqua's call to the **Campaign Status Notification URL** tells AwesomeApp that the campaign or program is active and ready to receive contacts or custom object records.

These calls are made using templated URLs to differentiate the call out. For example, the notification call for activating a campaign would resemble:

```
POST https://example.com/awesomeapp/feeders/notify?instance=9347bfe1-9c72-409c-a5cd-402ff74f0caa&asset=456&status=Activated
```

When the campaign or program is deactivated, Eloqua's call to the **Campaign Status Notification URL** tells AwesomeApp to stop sending contacts or custom object records. The deactivate notification would resemble:

```
POST https://example.com/awesomeapp/feeders/notify?instance=9347bfe1-9c72-409c-a5cd-402ff74f0caa&asset=456&status=Draft
```

After the campaign or program is activated, AwesomeApp responds with a 204 response, indicating that the response will be asynchronous, followed by an **import** to Eloqua's Bulk API, where AwesomeApp specifies a sync action. For AppCloud Feeders, the sync action sets the AppCloud Feeder instance's status to **complete**. Note that the maximum import size per batch is 5,000: if you have more than 5,000 records, break your data up into multiple batches.

Bulk API contact import

1. Create the bulk import definition, setting the status of the AppCloud Feeder instance to **complete**. The "destination" field refers to the AppCloud Feeders service's instance - in this example, the instance ID is `9347bfe1-9c72-409c-a5cd-402ff74f0caa`:

❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

Create a contact import definition where the sync action sets the record's status, where the instance ID is `9347bfe1-9c72-409c-a5cd-402ff74f0caa` :

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports
{
  "name" : "AwesomeApp Feeder Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "emailAddress" : "{{Contact.Field(C_EmailAddress)}}"
  }
  "syncActions" : [
    {
       "destination" : "{{FeederInstance(9347bfe19c72409ca5cd402ff74f0caa)}}",
       "action" : "setStatus",
       "status" : "complete"
    }
  ],
  "identifierFieldName" : "emailAddress"
}
```

❑ **Warning**: All field names and values in Eloqua data transfer objects are case sensitive. Be sure to reproduce them exactly. For example: `{{Contact.id}}` would fail if the correct form is `{{Contact.Id}}`.

Eloqua's response will be a 201 Created response that includes a `uri` parameter, which you can use to identify the import:


```
{
 "name": "AwesomeApp Feeder Bulk Import",
 "updateRule": "always",
 "fields": {
  "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "identifierFieldName": "emailAddress",
 "syncActions": [
  {
   "destination": "{{FeederInstance(9347bfe19c72409ca5cd402ff74f0caa)}}",
   "action": "setStatus",
   "status": "complete"
  }
 ],
 "isSyncTriggeredOnImport": false,
 "isUpdatingMultipleMatchedRecords": false,
 "uri": "/contacts/imports/6",
 "createdBy": "DocsExample",
 "createdAt": "2014-03-06T13:59:00.6600046Z",
```

```
    "updatedBy": "DocsExample",
    "updatedAt": "2014-03-06T13:59:00.6600046Z"
  }
```

2. Send Eloqua the import data using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/6/data
```

```
[
    {
      "emailAddress" : "fred@example.com"
    },
    {
      "emailAddress" : "sylvie@example.com"
    }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/contacts/imports/6"
}
```

   Eloqua's response:

```
201 Created
```

```
{
    "syncedInstanceURI" : "/contacts/imports/6",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

4. You can then use the sync's URI (`/syncs/6`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/6
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
    "syncedInstanceURI" : "/contacts/imports/6",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/6"
}
```

Bulk API custom object import

1. Create the bulk import definition, setting the status to yes. The destination field refers to the feeder service's instance - in this example, the instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`:

❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

Create a custom object import definition where the sync action sets the record's status, where the instance ID is `f82d50cd-86a9-4fca-b37e4ec9a98b0339`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/9/imports

{
  "name" : "AwesomeApp Feeder Response Bulk Import",
  "updateRule" : "always",
  "fields" : {
    "email" : "{{CustomObject[9].Field[58]}}"
  },
  "syncActions" : [
    {
      "destination" : "{{FeederInstance(f82d50cd86a94fcab37e4ec9a98b0339)}}",
      "action" : "setStatus",
      "status" : "complete"
    }
  ],
  "identifierFieldName" : "email"
}
```

Eloqua's response will be a 201 Created response that includes a `uri` parameter, which you can use to identify the import:

```
HTTP/1.1 201 Created

{
    "name" : "AwesomeApp Feeder Response Bulk Import",
    "updateRule" : "always",
    "fields" : {
      "email" : "{{CustomObject[9].Field[58]}}"
    },
    "identifierFieldName" : "email",
    "syncActions" : [
      {
        "destination" : "{{FeederInstance(f82d50cd86a94fcab37e4ec9a98b0339)}}",
        "action" : "setStatus",
        "status" : "complete"
      }
    ],
    "isSyncTriggeredOnImport" : false,
    "isUpdatingMultipleMatchedRecords" : false,
    "uri" : "/customObjects/imports/9",
    "createdBy" : "DocsExample",
    "createdAt" : "2014-03-06T13:59:00.6600046Z",
    "updatedBy" : "DocsExample",
    "updatedAt" : "2014-03-06T13:59:00.6600046Z"
}
```

2. Send Eloqua the data for import as a using the `uri`:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/customObjects/imports/9/data
```

```
[
    {
      "email" : "fred@example.com"
    },
    {
      "email" : "sylvie@example.com"
    }
]
```

3. Synchronize the data for import:

   AwesomeApp's request:

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
   "syncedInstanceURI":"/customObjects/imports/9"
}
```

   Eloqua's Response:

```
201 Created


{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "status" : "pending",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

4. You can then use the sync's uri (`/syncs/9`) to check the status of the sync:

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/9
```

When the sync is complete, Eloqua's response will resemble:

```
200 OK


{
    "syncedInstanceURI" : "/customObjects/imports/9",
    "syncStartedAt" : "2014-01-01T13:59:07.1375620Z",
    "syncEndedAt" : "2014-01-01T13:59:14.1375620Z",
    "status" : "success",
    "createdAt" : "2014-01-01T13:59:07.1375620Z",
    "createdBy" : "DocsExample",
    "uri" : "/syncs/9"
}
```

> **Learn more**: Bulk API imports.

Delete URL

The delete URL is a templated URL pointing to an endpoint for deleting an instance of your service.

The delete URL uses an HTTP DELETE request and there is no content sent in the request body. All common URL template parameters are available (the same as with a create URL). On success, this endpoint should return a 200-level response.

An example delete URL would look something like:

```
https://www.someurl.com/delete/{appId}/{installId}/{instanceId}/{userName}/
{siteName}/{siteId}
```

> **Note**: Delete calls are not sent in real time, but are done in a batch once daily.

Develop an Oracle Eloqua app menu service

The menus service is all about context. With menus, you can build an App that enables a marketer to launch an externally-hosted application from within Eloqua via a menu dock. This menu dock floats on the right side of the screen in the Eloqua's Campaign Canvas or the asset editor screens. When a marketer clicks on your menu app, Eloqua calls out to your service via the action URL, passing along context-specific data to your app.

Suppose AwesomeApp has a menu service. When a marketing user clicks on AwesomeApp, they are directed to the callout URL: `https://example.com/awesome/callout`. The call can include contextual information for your app to use. Available attributes include:

- `siteId` and `siteName`: these describe the Eloqua "site", generally the company or organization of the user
- `userId` and `userName`: the Eloqua username and id for the user who selected the menu service

- `userCulture`: the linguistic profile of the user. Eloqua uses Microsoft's CultureInfo class to define culture. For example, for English (United States), the code is `en-US`; for French (Canada), `fr-CA`
- `assetId` and `assetName`: the ID and name of the asset that the user was viewing when they opened the menu
- `assetType`: the type of asset, that the user was viewing when they opened the menu. Assets include account, campaign, contact, landing page, template, and so on.

The actual call to AwesomeApp would then resemble:

```
GET
https://example.com/awesome/callout?siteId=123&siteName=Eloqua&userId=1234&user
Name=Docs.Example&userCulture=en-
US&assetId=4&assetName=Docs%20Example%20Campaign&assetType=Campaign
```

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

Develop an Oracle Eloqua app firehose service

Eloqua's firehose service type enables third-parties to build a service that receives a notification when a marketer performs an action in Eloqua. When a marketer creates a campaign or edits an email, for example, Eloqua can call out to the third-party with the update.

Firehose supports subscriptions to a wide variety of events for the different asset types:

- **Email**: Created, Updated, Deleted
- **LandingPage**: Created, Updated, Deleted

- **ContactSegment**: Created, Updated, Deleted
- **Form**: Created, Updated, Deleted
- **Campaign**: Created, Updated, Deleted, Draft, DraftApproved, DraftWaitingApproval, ActivatedBySchedule, CampaignLandingPageAdded, CampaignEmailAdded, CampaignFormAdded, CampaignLandingPageRemoved, CampaignEmailRemoved
- **Program**: Created, Updated, Deleted, Draft, Activated, Paused

When you configure your app, you specify which events to subscribe to in the pattern. Then, when a marketer triggers one of those events, Eloqua calls out to the configured **notification URL** with the data specified in the firehose pattern.

For example, AwesomeApp has a firehose service and is configured to request all campaign-related events. The call to its notification URL would then resemble:

```
POST https://example.com/firehose
{
   "siteId": "123",
   "assetId": "9999",
   "assetType": "Campaign",
   "eventDate": "12/12/2014 12:00:01 AM",
   "eventType": "Updated",
   "userId": "1234",
   "userName": "Docs.Example",
   "msgAttributes": {
      "name": "name_of_updated_asset",
      "statuschangedby": "1234",
      "statuschangedat": "12/12/2014 12:00:00 AM"
   },
   "oath_consumerkey": "stuff"
}
```

Eloqua signs all outgoing calls with OAuth 1.0a so the receiving system can validate that the call was sent by Eloqua. Refer to the OAuth 1.0a spec or OAuth 1.0 RFC for more information.

> **Note**: If a firehose service is configured to receive form-related or segment-related events, app providers should expect to see two events whenever a form or segment is created or saved. When a form is created, Eloqua sends a `POST` request (for the creation) and a `PATCH` request (for the update). App providers should expect to see two events in these scenarios.

Register your app

When you have developed an app and are ready to start testing it, you need to register it with Eloqua. By now, you will have already registered as an app provider in your Eloqua development instance. Now, you need to register your app and services.

Once registered, you can test your apps within your instance to make sure they work as expected, and release the apps selectively or to the general public.

Step 1: Register the app

To register your app, select **Settings** > **AppCloud Developer**.

If you have not yet registered any apps, you will see a “You currently have no apps” message, and the **Create App** button.

If you have already registered an app, it will be listed alongside the **Create App** button.

To register your app, click **Create App**.

App information

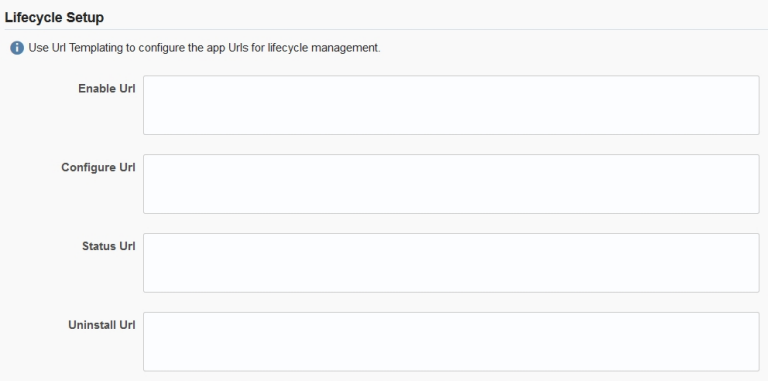
The *Create App* screen prompts you to fill in a number of fields that describe your app. Complete these fields with the relevant information. Current fields include **Name**, **Description**, and **Icon**.

> ❑ **Warning**: the **Name** field is limited to 100 characters and the **Description** field is limited to 4000 characters.

Lifecycle fields

- **Enable URL**:The URL called when the App is first installed or when it is reconfigured. This is a templated URL which should include, at a minimum, the {InstallId}, {AppId}, and {CallbackUrl} parameters. WIthout the callback URL you will not be able to complete the installation flow if your app requires configuring (for example, to use OAuth).
- **Configure URL**: The URL called when the app is configured.
- **Status URL**:The URL called when a user checks the App's connection. Learn how to respond when Eloqua calls the Status URL.

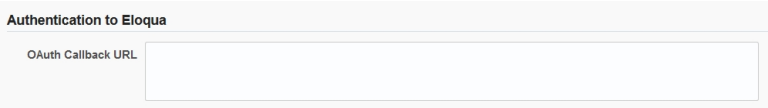
- **Uninstall URL**: The URL called when a user uninstalls the app. This allows Eloqua to notify the app provider when someone uninstalls their app from the appcloud catalog. This URL supports templated parameters such as `{InstallId}`, `{AppId}`, etc.



Authentication to Eloqua

Apps for the AppCloud use OAuth for authentication.

- **OAuth Callback URL**: the URL that the users should be redirected to after installing the app and authenticating with Eloqua.



To use OAuth, you'll need the following information which is displayed after you click **Save**:

- **Client ID (app ID)**: Eloqua creates and assigns the app ID when you save your app.
- **Client Secret**: Eloqua generates the secret access token when you save your app. Click Show to view it.

Once you complete all the fields, click **Save** to finish configuring the app, or click **Add Services** to start registering services.

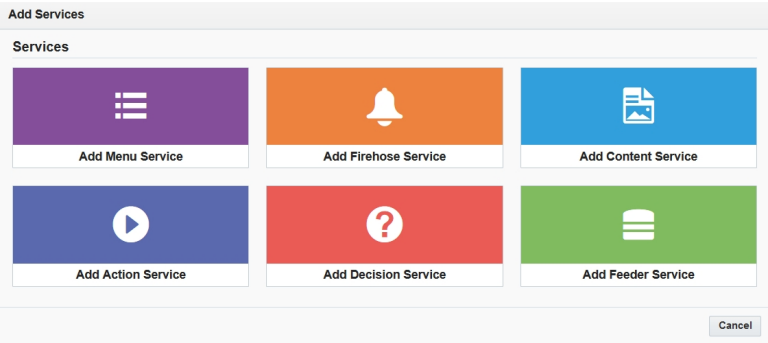
Step 2: Register services

Finally, you should register any service your app uses. The AppCloud developer platform currently supports 6 services: menus, firehose, content, actions, decisions, and feeders.

Apps can include zero services, as in the case of a portal app or if the integration not directly tied to one of the service types, or if it is tied to many services.

To define your app's services within Eloqua, navigate to **Settings** > **AppCloud Developer** then click on your app (see step 1 above for setup instructions) and select **Add Services**.

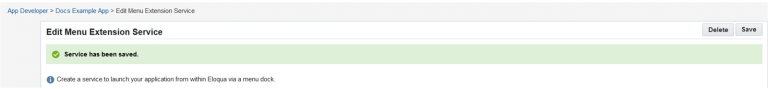
A modal opens listing the available service types. Select the service type you wish to register.



Then, fill out the service's details: you will have different information you need to supply depending on the service you are implementing. For example, for a menu, you need to provide the action URL, specify the areas it supports and how Eloqua should present it, and provide the service's icon URL, name, and description; for content, you will need to fill out instance configuration details, as well as configure content and notification settings, as well as the general service details. The following documents describe the service configuration page for each service type:

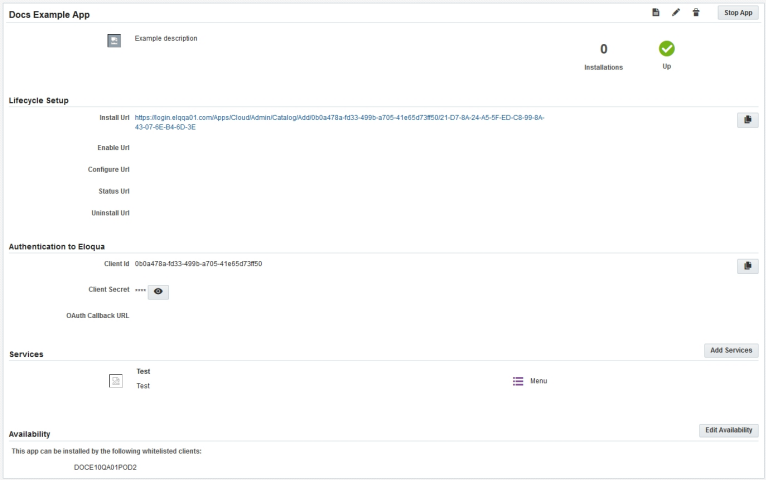
- Menus
- Content
- Actions
- Decisions
- Feeders
- Firehose

When you are done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.



Step 3: View the app

You can view the app information by clicking its name in the list. Here is an example of the *App Details* page for *Docs Example App*:



The *App Details* page shows all of the app information, including *Services*. When you register an app, it immediately becomes available in your instance of Eloqua so you can test it before releasing it to any other prospective users. When you are ready to release your app, or just to share it with a select few, check out the app publishing tutorial.

Register an action service

With sctions, developers can build apps that control an external action that occurs when contacts enter a step on the campaign canvas. Examples of this could include sending contacts SMS messages, a direct mail piece, or registering them for webinars. When contacts enter an action step, Eloqua will call out to the app, and the app can respond by performing an action in an external system, then informing Eloqua that the process is complete. The contact will then move on in the workflow.

Actions are the next generation of Eloqua's cloud connectors framework.

When you set up an action service, you will need to provide the usual Oracle Eloqua app Developer framework service details, as well as the instance configuration and action settings.

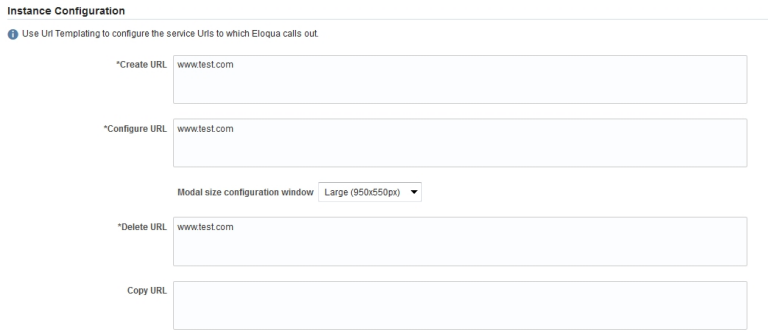
Service Details

These fields are present for all services:

- **Name**: the name of the service. Max length: 100 characters.
- **Description**: describes what the service does. Max length: 4000 characters.
- **16x16px Icon**: the URL of the icon Eloqua displays when your service is displayed on a canvas. The icon will be 16px by 16px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.
- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The icon will be 32px by 32px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.

Instance configuration

This section defines the URLs for creating, configuring, copying, and deleting an instance of your service. The installation tutorial details the instance creation flow when a marketer drags an Action step onto the canvas, and selects your app.
Each URL is a *templated URL* that uses common URL template parameters and some Eloqua markup language parameters. Eloqua replaces these paramters with their appropriate values when it makes a call. For more about URL templating, see our Introduction to URL templating.



- **Create URL**: A templated URL pointing to a web portal for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future. On success, this endpoint should return a 200-level response with the created instance.

- **Configure URL**: A templated URL pointing to an endpoint for configuring an instance of this service as an HTTP `GET` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future.

- **Modal size configuration window**: Select a modal size for your configuration window. The configuration page displays when marketers configure your service.

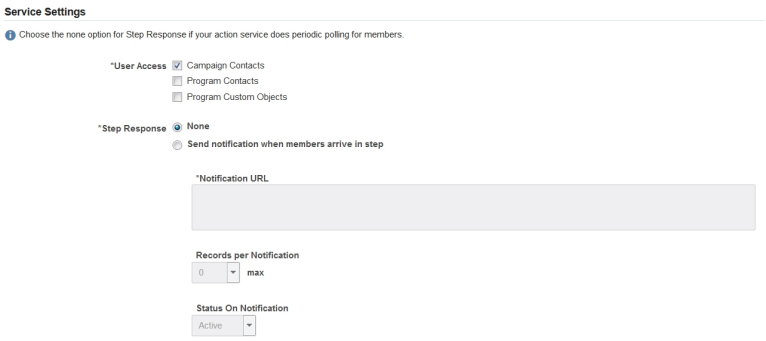
  - **Large**: 950px x 550px
  - **Small**: 650px x 550px

- **Delete URL**: A templated URL pointing to an endpoint for deleting an instance of this service using an HTTP `DELETE` request. All common URL template parameters are available. On success, this endpoint should return a 200-level response.

- **Copy URL**: A templated URL pointing to an endpoint for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available in addition to `{OriginalInstanceId}`, `{OriginalInstallId}`, `{OriginalAssetId}`, and `{OriginalAssetName}`.

  You should be sure to include, at a minimum, the `{InstanceId}` and `{OriginalInstanceId}` parameters so that you can identify the original and newly created instances. On success, this endpoint should return a 200-level response with the created instance. Learn how to Respond when a marketer copies a service instance.

Service settings



- **User access**: Specify if your service supports campaign contacts, program contacts, and/or program custom objects.
- **Step response**: Specify whether Eloqua should call out to your service when a member arrives in an Action Step, or whether you will poll for members periodically.
- **Notification URL**: This url will be called (HTTP POST) when actions are taken on an instance of this service. This endpoint should return a 2xx level response with an empty body on success. If this property is not specified, the system will fall back to a polling style approach without notification. Note that this url can be used along with the `recordDefinition` property of the instance to send data.
- **Records per notification**: Max number of records to push per HTTP request (between 0 and 5,000). If set to 0, apps must be developed to retrieve records, see the tutorial Retrieving app records using the bulk API for more information.
- **Status on Notification**: Specifies what the member's status should be set to when a notification is set. This is only valid if *max records per notification* is greater than 0. If you set the *status on notification* to **Complete**, Eloqua will call out to the notification URL when contacts flow into the action step, and will then push the contacts directly into the next step. If you set it to **Active**, you will need to call back into Eloqua to set each contact's status to complete so they will flow into the next step. See: develop an action service for more information.

When you're done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

Register a Decision Service

With Decisions, developers can build apps that directly control the path a contact takes as they flow through a campaign. For instance, you could build an App that Eloqua calls out to when contacts reach a Decision step. This App might interact with an external system, and evaluate rules against that set of external data. The App then responds, telling Eloqua whether the contacts should flow through the step's "yes" or "no" path. With Decisions, the data used in the decision never needs to be brought into Eloqua.

When you set up a Decision service, you will need to provide the usual AppCloud Developer framework service details, as well as the **Instance Configuration**, **Content-Type of the notification POST call** and **Decision Settings**.

Service Details

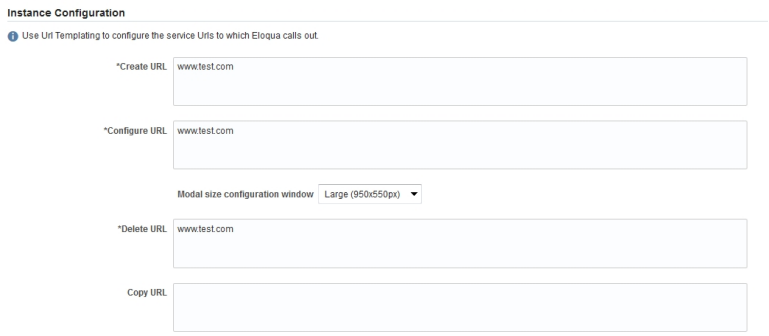
These fields are present for all services:

- **Name**: the name of the service. Max length: 100 characters.
- **Description**: describes what the service does. Max length: 4000 characters.
- **16x16px Icon**: the URL of the icon Eloqua displays when your service is displayed on a canvas. The icon will be 16px by 16px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.
- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The icon will be 32px by 32px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.

Instance Configuration

This section defines the URLs for creating, configuring, copying, and deleting an instance of your service. The installation tutorial details the instance creation flow when a marketer drags an Decision step onto the canvas, and selects your app.

Each URL is a *templated URL* that uses common URL template parameters and some Eloqua markup language parameters. Eloqua replaces these paramters with their appropriate values when it makes a call. For more about URL templating, see our Introduction to URL templating.



- **Create URL**: A templated URL pointing to a web portal for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future. On success, this endpoint should return a 200-level response with the created instance.

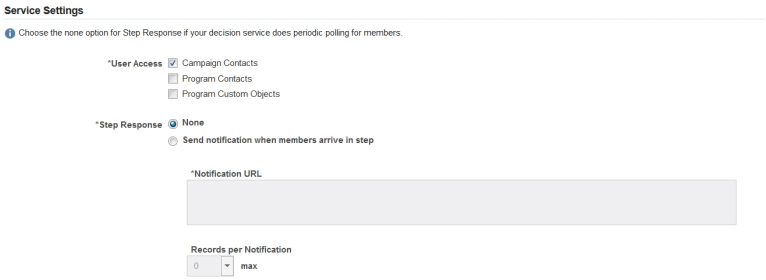
- **Configure URL**: A templated URL pointing to an endpoint for configuring an instance of this service as an HTTP `GET` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future.

- **Modal size configuration window**: Select a modal size for your configuration window. The configuration page displays when marketers configure your service.
    - **Large**: 950px x 550px
    - **Small**: 650px x 550px
- **Delete URL**: A templated URL pointing to an endpoint for deleting an instance of this service using an HTTP `DELETE` request. All common URL template parameters are available. On success, this endpoint should return a 200-level response.
- **Copy URL**: A templated URL pointing to an endpoint for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available in addition to {OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, and {OriginalAssetName}.

  You should be sure to include, at a minimum, the {InstanceId} and {OriginalInstanceId} parameters so that you can identify the original and newly created instances. On success, this endpoint should return a 200-level response with the created instance. Learn how to Respond when a marketer copies a service instance.

Service Settings



- **User Access**: Select the Eloqua services for which your app should be available.
- **Step Response**: Specify whether Eloqua should call out to your service when a member arrives in the decision step, or whether your application will poll for members periodically.

- **Notification URL**: A templated URL to an endpoint for notifying the external service, during execution, to get the decision result of the configured instance. If this property is not specified, the system will fall back to a polling style approach without any notification.
- **Records per Notification**: Max number of records to push per HTTP request (between 0 and 5,000). If set to 0, apps must be developed to retrieve records, see the tutorial Retrieving app records using the bulk API for more information.

When you're done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

Register a content service

Content allows marketers to source pieces of content in Eloqua emails and landing pages, from an external source. This is the new and improved version of Eloqua's cloud components framework, and it includes many improvements such as asynchronous bulk processing (for emails), the ability to fully test the content service within Eloqua, and design-time interaction with the email and landing page editors. For example, you can now resize content in real time.

When you set up a content service, you will need to provide the usual AppCloud developer framework service details, as well as the **Instance Configuration** and **Content Settings**.

Service Details

These fields are present for all services:

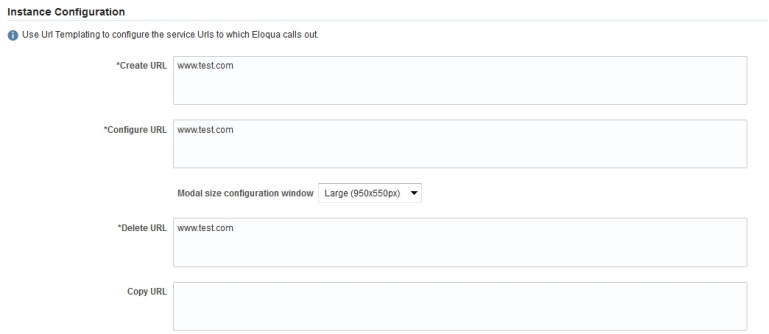
- **Name**: the name of the service
- **Description**: describes what the service does

- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The URL must be `https://`

Instance configuration

This section defines the URLs for creating, configuring, copying and deleting an instance of your service. The installation tutorial details the instance creation flow when a creates a landing page or email populated with content.

Each URL is a *templated URL* that uses common URL template parameters and some Eloqua markup language parameters. Eloqua replaces these paramters with their appropriate values when it makes a call. For more about URL templating, see our Introduction to URL templating.



- **Create URL**: A templated URL pointing to a web portal for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future. On success, this endpoint should return a 200-level response with the created instance.

- **Configure URL**: A templated URL pointing to an endpoint for configuring an instance of this service as an HTTP `GET` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future.

- **Modal size configuration window**: Select a modal size for your configuration window. The configuration page displays when marketers configure your service.

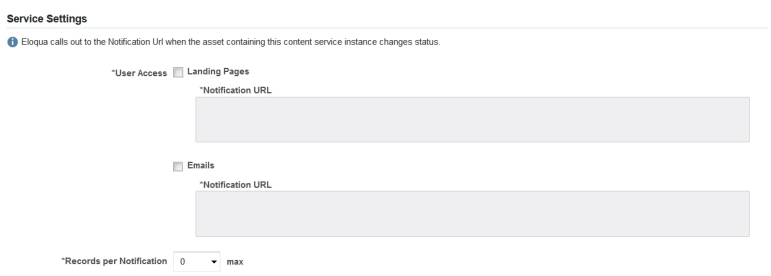
  - **Large**: 950px x 550px
  - **Small**: 650px x 550px

- **Delete URL**: A templated URL pointing to an endpoint for deleting an instance of this service using an HTTP `DELETE` request. All common URL template parameters are available. On success, this endpoint should return a 200-level response.

- **Copy URL**: A templated URL pointing to an endpoint for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available in addition to `{OriginalInstanceId}`, `{OriginalInstallId}`, `{OriginalAssetId}`, and `{OriginalAssetName}`.

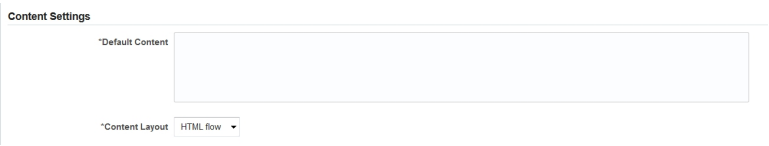
  You should be sure to include, at a minimum, the `{InstanceId}` and `{OriginalInstanceId}` parameters so that you can identify the original and newly created instances. On success, this endpoint should return a 200-level response with the created instance. Learn how to Respond when a marketer copies a service instance.

Service settings



- **User Access**: Select the Eloqua asset types for which your app should be available.
    - **Landing Page Notification (HTML) URL**: templated URL that Landing Page engines should call out to during render to retrieve the HTML content for the configured instance. In addition to the common HTML template parameters, the notification (HTML) URL supports the {VisitorId}, `{Fragment}`, `{Host}`, `{Query}` and `{Url}` parameters. Be sure to include, at a minimum, the the {InstanceId} and {ExecutionId} parameters so that you will be able to call back in to Eloqua.
    - **Email Notification (HTML) URL**: templated URL that the email engine should call out to during render to retrieve the HTML content for the configured instance. Be sure to include, at a minimum, the the {InstanceId} and {ExecutionId} parameters so that you will be able to call back in to Eloqua.
- **Max Records per Notification**: max number of records to push per HTTP request (between 0 and 5,000)

Content settings



- **Default Content**: the HTML content the system should use if the external service is not accessible. This is a required field and cannot be empty.
- **Content Layout**: choose from flow or fixed. If you choose flow, the external content is input into your landing page or email with no constraints (except width). If you choose fixed, the external content will be constrained in height and width, as specified by the content instance, and will not be configurable from within Eloqua.

If the landing page or email notification URL is not specified and a landing page or email rendering is required, the default content is used instead.

When you're done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

Register a feeder service

With feeders, developers can build apps that use data in third-party services to drive campaign membership. When you set up a feeder service, you will need to provide the usual AppCloud developer framework service details, as well as the **Instance Configuration** and **Feeder Settings**.

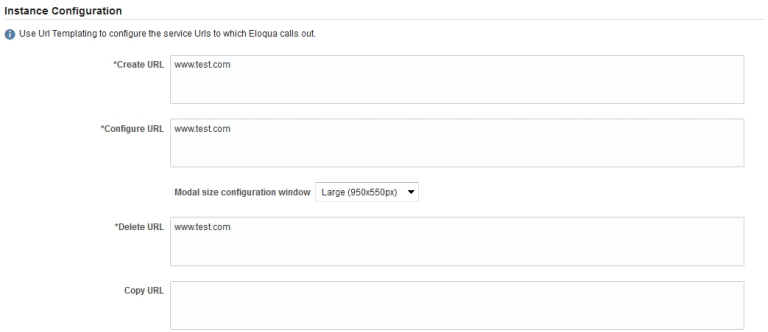
Service Details

These fields are present for all services:

- **Name**: the name of the service. Max length: 100 characters.
- **Description**: describes what the service does. Max length: 4000 characters.
- **16x16px Icon**: the URL of the icon Eloqua displays when your service is displayed on a canvas. The icon will be 16px by 16px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.
- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The icon will be 32px by 32px when displayed. It must be a secure URL. See App icon design guidelines for more information on designing app icons.

Instance configuration

This section defines the URLs for creating, configuring, copying, and deleting an instance of your service. The installation tutorial details the instance creation flow when a marketer drags an feeder step onto the canvas, and selects your app.
Each URL is a *templated URL* that uses common URL template parameters and some Eloqua markup language parameters. Eloqua replaces these paramters with their appropriate values when it makes a call. For more about URL templating, see our Introduction to URL templating.



- **Create URL**: A templated URL pointing to a web portal for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future. On success, this endpoint should return a 200-level response with the created instance.

- **Configure URL**: A templated URL pointing to an endpoint for configuring an instance of this service as an HTTP `GET` request. All common URL template parameters are available.

  You should be sure to include, at a minimum, the `{InstanceId}` parameter so that you will be able to identify the service in the future.

- **Modal size configuration window**: Select a modal size for your configuration window. The configuration page displays when marketers configure your service.

  - **Large**: 950px x 550px
  - **Small**: 650px x 550px

- **Delete URL**: A templated URL pointing to an endpoint for deleting an instance of this service using an HTTP `DELETE` request. All common URL template parameters are available. On success, this endpoint should return a 200-level response.

- **Copy URL**: A templated URL pointing to an endpoint for creating an instance of this service as an HTTP `POST` request. All common URL template parameters are available in addition to

{OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, and {OriginalAssetName}.

You should be sure to include, at a minimum, the {InstanceId} and {OriginalInstanceId} parameters so that you can identify the original and newly created instances. On success, this endpoint should return a 200-level response with the created instance. Learn how to Respond when a marketer copies a service instance.

Feeder settings

- **User Access**: Select the Eloqua services for which your app will be available.
- **Notification URL**: This URL is called (HTTP `POST`) when a campaign containing an instance of the service's status changes, such as when it is activated.

  Be sure to include the {AssetId} and {EventType} parameters in the templated URL. `AssetId` describes the campaign whose status has changed, and `EventType` defines the status. When a campaign is activated, for example `EventType` will be “Active”. Otherwise, you will not be able to identify what the incoming call references.

When you’re done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

Register a menu service

Eloqua's menus service is all about *context*. With menus, you can build an App that enables a marketer to launch an externally-hosted application from within Eloqua via a menu dock. This menu dock floats on the right side of the screen in the Eloqua's campaign canvas or the asset editor screens. When a marketer clicks on your menu

app, Eloqua calls out to your service on the specified endpoint URL, passing along information about the asset or area from which your App was launched.

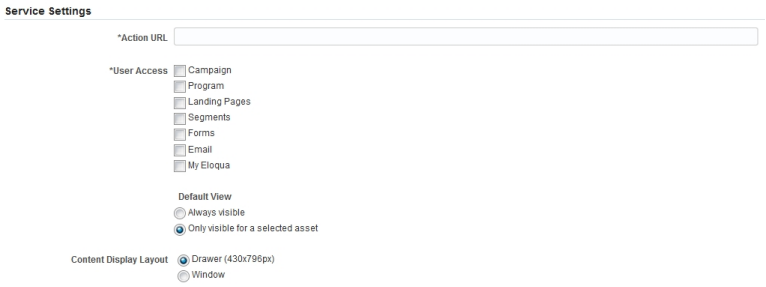
When setting up a menu service, you will need to provide basic service details, as well as the **Menu Extension Service Details**.

Service Details

These fields are present for all services:

- **Name**: the name of the service
- **Description**: describes what the service does
- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The URL must be `https://`

Service settings



- **Action URL**: A templated URL that defines where users are redirected to when they select the menu item. For example, a new tab with an external page showing information on a Campaign.
- **User Access**: Select the assets/areas within Eloqua that your app will appear to the Marketer.

- **Default View**: Specify how your service will appear.
    - **Always visible**: Display the menu service in two areas: within the selected area of the interface and within the editor for the asset.

      Example: If this option is selected along with *Campaign* access, a user will see the menu service when they navigate to **Orchestration** > **Campaigns**, and when they open a campaign.

    - **Only visible for a selected asset**: Display the menu service only for the selected assets under user access.

      Example: If this option is selected along with *Campaign* access, a user will only see the menu service when they open a campaign.

- **Content Display Layout**: Select a layout option for your content display.
    - **Drawer**: Content will open within the AppCloud menu dock.
    - **Window**: Content will open in a new window or tab.

When you are done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

Register a Firehose Service

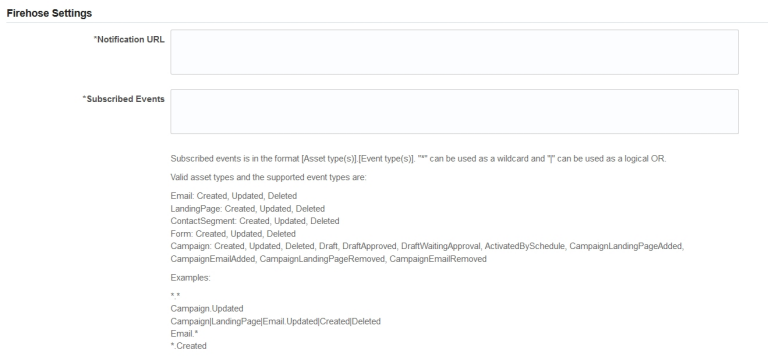
Firehose enables third-parties to build a service that receives a notification when a marketer performs an action in Eloqua. This is essentially a web-hook that will be triggered when a marketer does things like creating or editing a campaign, or creating or updating an email etc.

Service Details

These fields are present for all services:

- **Name**: the name of the service
- **Description**: describes what the service does
- **32x32px Icon**: the URL of the icon that Eloqua should display for your service. The URL must be `https://`

Firehose Settings



- **Notification URL**: the URL that Eloqua will call out to with the notification. This is a templated URL.
- **Subscribed Events**: defines which events trigger a call out to the external application. The service configuration page includes a list of valid asset types and the event types supported for each asset, as well as examples of valid patterns.

When you're done configuring your service, click **Save**. A green alert message will appear at the top of the page indicating the service has been successfully saved.

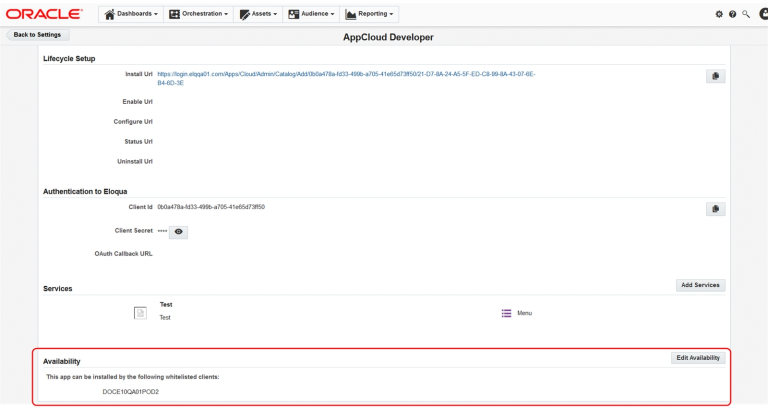
Publish your app

The final step in the app configuration lifecycle is publishing your app. The publishing flow has two components: whitelisting users so that they are permitted to install the app, and retrieving the link that those users need in order to add the app to their catalog.

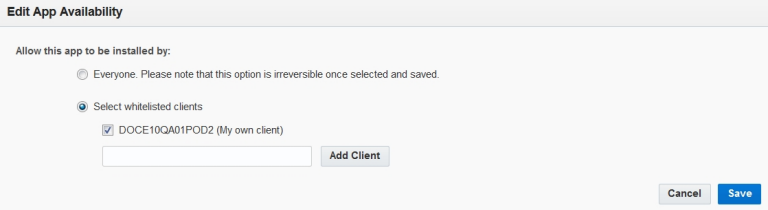
Grant access

By default, when you register an app with Eloqua, only your instance of Eloqua has access to it. This makes it easy for you to test your app to ensure it is behaving as expected.

To allow other people access to the app, you have to grant access from the app Configuration page, by clicking **Edit Availability** in the **Availability** section.



You can choose between granting access to all Eloqua instances, or select Eloqua instances.



Publish to all sites

If your app is available to all Eloqua instances, all sites are immediately whitelisted. At present, this cannot be undone. Nevertheless, a customer admin will need the install URL before they can actually install the app.

Publish to a specific site

To make your app available to specific Eloqua instances, enter the name of the site you want to add and click **Add Client**. All Eloqua Instances with that site name will now be able to install the app.

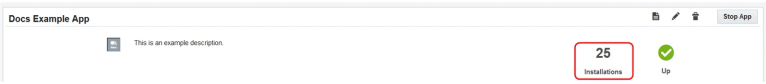
> **Note**: You cannot remove a site from your app's whitelist if that site has your app installed. To remove a site from your app's whitelist, you must first contact the appropriate client and have them uninstall your app. This a security measure in place to prevent broken assets and campaigns.

Share the URL

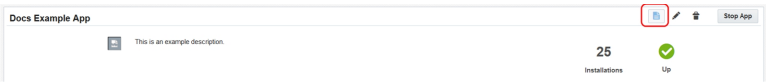
Having published your app, giving sites the ability to install it, you now need to share the URL so that users can add the app to their Eloqua instance's app Catalog, giving marketers the ability to use it. Eloqua automatically creates the installation URL when you register your app.

Many developers send the link by email to the sites they have authorized. When an authorized user clicks the link, the app is added to their catalog. The link will not work for users who are not authorized, so you do not need to worry about people passing around the installation link to sites you did not whitelist.

The number of sites which have installed your app is listed in the app details page.



Click **Logs** to see catalog install details.



Next steps

When a customer admin installs an app on a new site, Eloqua calls out to the app's install URL. Refer to the AppCloud Installation Flow tutorial for more information. You may also want to publish your app to the Oracle Cloud Marketplace.

AppCloud installation flow

When a Customer Admin installs an app, Eloqua calls out to the configured **enable URL** with a POST request. The call will resemble the following:

```
https://example.com/awesomeapp/api/enable?enableurl&oauth_consumerkey=XXX-XXX-XXX-XXX-XXX
```

```
&oauth_nonce=nonce&oauth_signature_method=SIG-METH&oauth_timestamp=111111111
&oauth_version=1.0&oauth_
signature=signature&callback=https%3A%2F%2Fsecure.eloqua.com%2FApps%2FCloud%2
FAdmin%2FInstall%2FCallback%2Fa1b2c3d4%3Fguid%3Dz9y8x7w6
```

The app must respond in one of two ways: if the app does not require configuring by the user, it can return an HTTP 200 OK response. Otherwise, it should respond with an HTTP 302 redirect to another page. The following sections detail each option.

No additional information needed

In this scenario, there is no need to obtain OAuth authorization for the installing user's instance, nor any other configuration information. The appropriate response is then:

```
HTTP/1.1 200 OK
```

The app install status will be set to the "Ready" and marketers will be able to use it.

Additional information needed

If you need to have the user configure the application or provide OAuth authorization for their instance, you should respond with an HTTP 302 redirect to another page. When the app is configured, you need to call back in to the callback URL.

1. Respond with a redirect:

   ```
   HTTP/1.1 302 Redirect
   ```

   If your app needed to go through the OAuth flow, the redirect might point to `login.eloqua.com/auth/oauth2/authorize`, Eloqua's OAuth endpoint. If you needed users to configure options in the app when they install it, the redirect might point to a configure page within your app.

2. Call in to the **callback URL** when the user has finished configuring the application. Using the above example, the call would look like:

   ```
   https://secure.eloqua.com/Apps/Cloud/Admin/Install/Callback/a1b2c3d4?guid=z9y8x7w6
   ```

   Eloqua provides the **callback URL** when you request it as a template in your templated **enable URL**. Calling the **callback URL** tells Eloqua that the install status should be set to "Ready" so that marketers can start using the application.

3. When the app provider calls back in to complete the installation, it can optionally append a redirect URL:

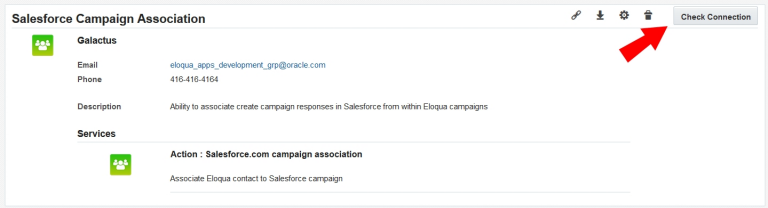
   ```
   https://secure.eloqua.com/Apps/Cloud/Admin/Install/Callback/{installId}?guid={guid}&redirect={redirectUrl}
   ```

   Replace parameters in {} with desired values, and the redirect URL must be in HTTP or HTTPS. The redirect URL supports templated URL parameters. You must URL encode the redirect URL to ensure that template parameters are returned back to you. If no redirect URL is given, or if the redirect URL given is not acceptable, then the user is redirected to Eloqua's catalog.

Respond when Eloqua calls the status URL

When you register an app in Eloqua, you are able to specify a status URL. This URL is an endpoint which Eloqua can call to verify that your app is up and running.

Eloqua calls the status URL when a user clicks the **Check Connection** button for that app from the AppCloud catalog.



Important: Applications have an application status associated with them. This status is either up, down, or maintenance. Calls to the status URL do not update the application status: they merely display the application's response to the user who clicked the **Status** button.

Example

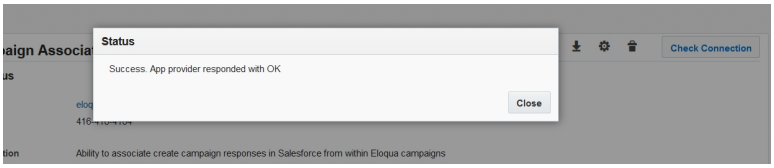
Suppose the user clicked the **Check Connection** button for AwesomeApp. The call resembles:

```
GET https://example.com/awesomeapp/status
```

If AwesomeApp is operating normally, the appropriate response is an 200-level response:

```
HTTP/1.1 200 OK
```

Eloqua then displays the status to the user in a modal window.



Otherwise, if AwesomeApp is down and responds with a 400 or 500-level message, Eloqua indicates that it was unable to connect to the application. Or that the application is in maintenance mode.

Respond when a marketer copies a service instance

Marketers often create new campaigns, landing pages, and emails from existing resources. When a marketer creates a campaign, landing page, or email using the **Save As** option in the campaign canvas and landing page or email editors all of the objects in that resource are duplicated. For most items on a canvas, this is straightforward. For services that follow the instantiation-execution model, however, each instance has its own unique identifier. Thus, when an action, decision, feeder, or content service instance is copied, a new GUID needs to be assigned to the copy. The Oracle Eloqua app developer framework uses the **Copy URL** to communicate with the service's provider to configure and update the new service instance when a service is copied.

Sample copy URL call and response

Eloqua's call to the copy URL is similar to the call to the create URL when a service is first created. The following section provides step-by-step instructions for properly handling calls to the copy URL. This example uses a decision service, but the process is the same for actions, content, and feeders.

1. A marketing user copies a campaign that includes an AwesomeApp decision service whose templated copy URL is:

```
example.com/awesomeapp/decide/copy?instanceId={InstanceId}&originalId=
{OriginalInstanceId}
```

2. Eloqua calls out to the copy URL, replacing the templated parameters with the original service's GUID, and the new GUID of the copy. AwesomeApp can use the original service's GUID to determine the appropriate configuration options for the copy:

```
POST example.com/awesomeapp/decide/copy?instanceId=bcbdff6a-74dd-41c9-
b716-825e4049b776
&originalId=d155213e-cd30-422b-984c-acb33093cb5b
```

3. AwesomeApp responds in one of two ways: either it replies with an HTTP 200 Ok response, and no content, in which case Eloqua will use the original service's configuration for the new service, or, AwesomeApp replies with a 200-level HTTP response along with a DTO specifying how the new service should be configured, as in the following:

```
HTTP/1.1 200 OK
{
  "recordDefinition":
    {
      "ContactId": "{{Contact.Id}}",
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
    }
}
```

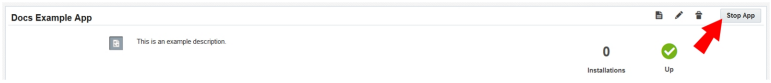
Internally, Eloqua then updates the references in the campaign to the new service instance GUID, and the new campaign is created.

As an app provider, you have control over what data and settings are persisted when a service is copied: if you wish to retain the configuration of the original service, simply return a blank DTO when Eloqua calls out to the copy URL. Conversely, if you wish to update the configuration of the new service, you can do so by replying with a new DTO.

Once a service has been copied, and the new service has been configured, there is ***no link*** between the original and new services: changing the configuration in one does not affect the other.

Scheduled maintenance

If you need to perform maintenance on your app, and wish to put it into maintenance mode, you can do so directly from the app's *App Details* page from within your Eloqua instance by clicking the **Stop App** button.



When an app is in maintenance mode:

- Eloqua will not send any calls to the app.
- Marketers should not be able to add the app's services to a campaign, email, or landing page.
- When a user checks the app's status, Eloqua will indicate the app is undergoing maintenance.

App shutdown is not equivalent to deactivating or deleting an app. It is useful if you wish to suspend the app while performing an upgrade.

App shutdown

If your app is not operating properly, Oracle Eloqua can shut down your app. This prevents marketers from creating new campaigns that use your app's services. Marketers can also configure a default path for contacts to follow in the event that a service is unavailable while a campaign is in progress. See: configuring cloud action and decision elements in the Eloqua user help for more information. When an app is shutdown, any contacts in the app's service's steps will flow down the default path, if it is configured.

Shutdown

Eloqua will shut down your app if there were **more than 100 calls in the last five minutes, and 25% of them failed**. Eloqua will automatically send an email message to the email address associated with your provider, with a link you can use to reactivate your app. When the app is shut down, all pending and future requests are marked as "failed".

Reactivate your app

To reactivate your app, click on the link in the notification email that Eloqua sent. Alternatively, log in to your development instance and click **Start App** on the *App Details* page of Eloqua's *AppCloud Developer Framework* area.

You can programatically start or stop your app using the `GET /apps/{appid}/configurations` and `PUT /apps/{appid}/configurations` endpoints. See: programmatically update your app's status for more information.

App icon design guidelines

When designing your app's icon, it is important to follow the recommended design guidelines to ensure your app icon displays correctly for your users. Each app requires:

- AppCloud catalog icon
- App-specific icon (the size requirement depends on where the app appears in the Eloqua interface)
    - Menu
    - Content
    - Firehose
    - Canvas

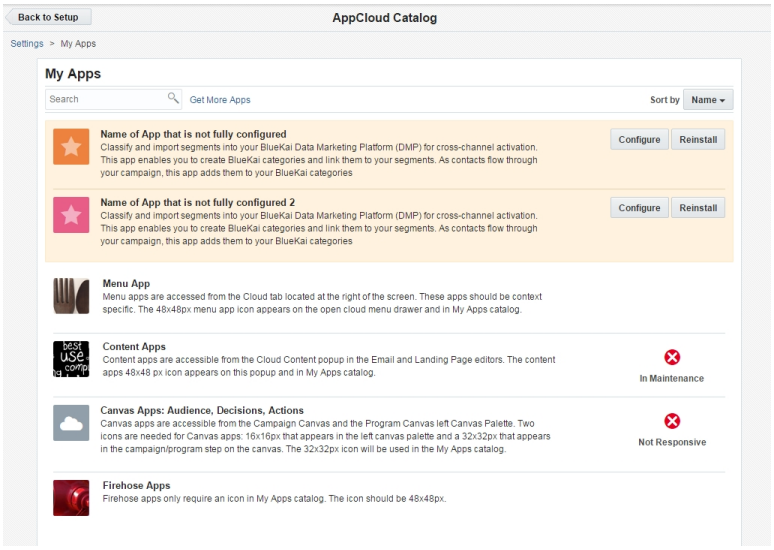
Each app has different size guidelines and will appear in different areas of the Eloqua interface:

AppCloud catalog icons

Size requirement:

- 96x96px

An icon to represent your app will be displayed in the AppCloud catalog and on the *App Details* page.

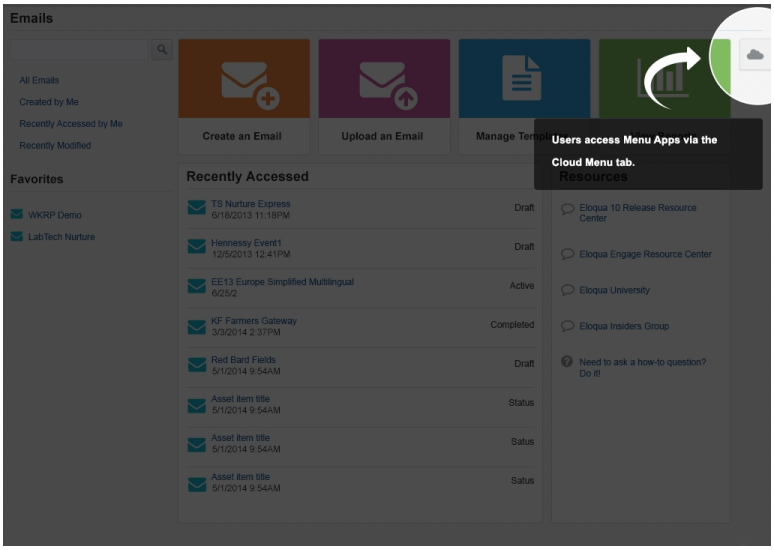


Menu apps

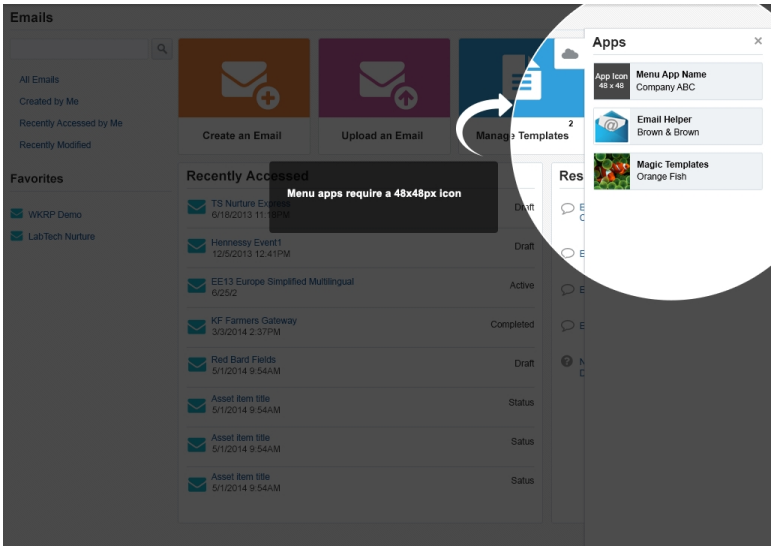
Size requirement:

- 96x96px

Menu apps are accessed from the cloud menu tab, located at the top right of the screen. These apps should be context specific.



The menu app icon appears within the cloud menu tab and in AppCloud catalog.

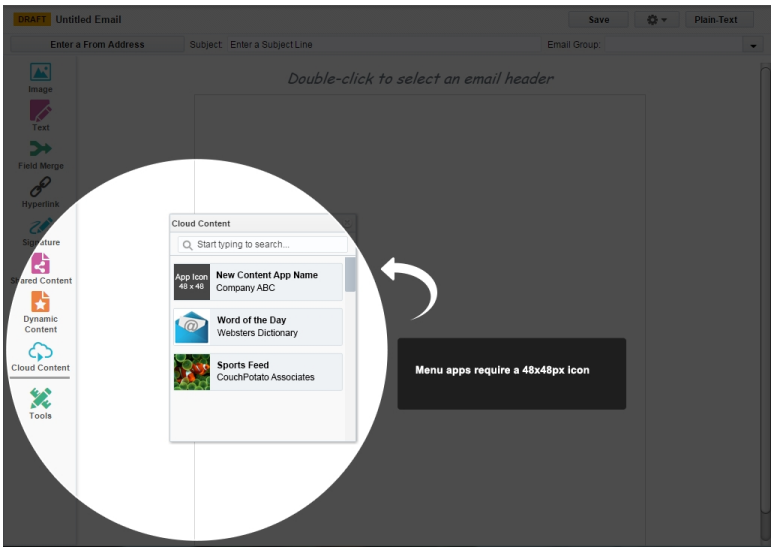


Content apps

Size requirement:

- 96x96px

Content apps are accessible from the cloud content popup in the email and landing page editors. The content apps icon appears on this popup and in the AppCloud catalog.

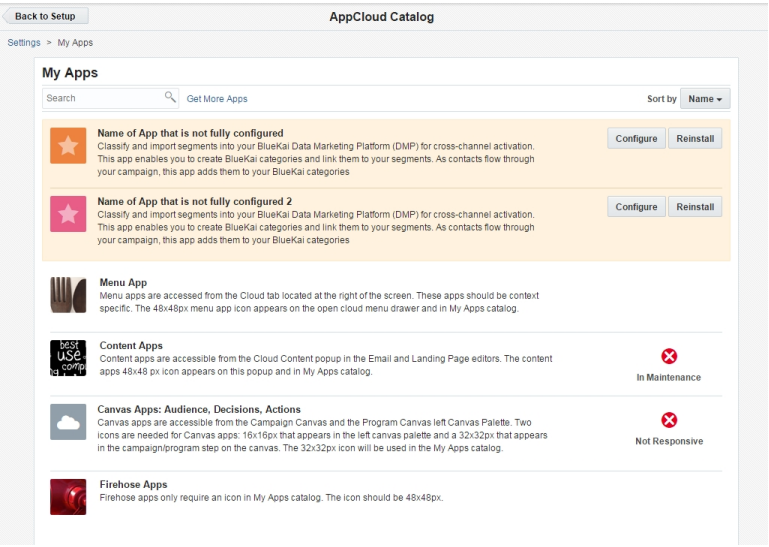


Firehose apps

Size requirement:

- 96x96px

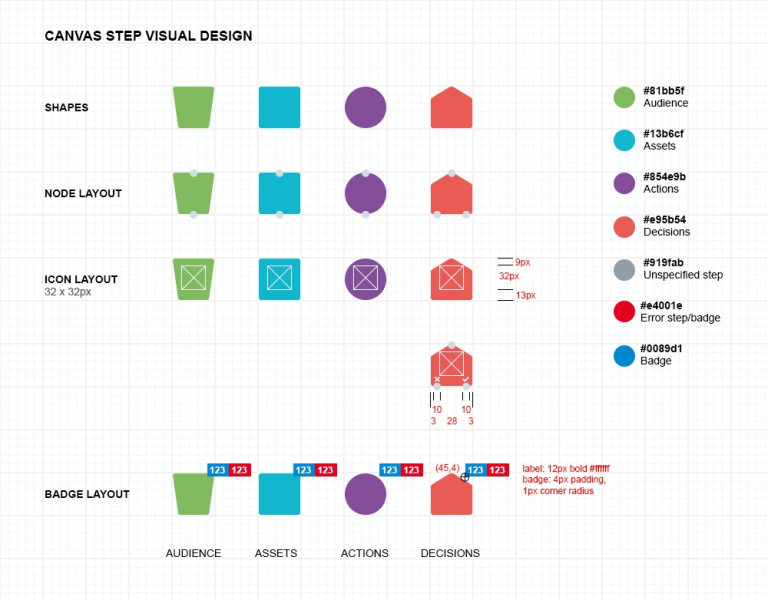
Firehose apps only require an icon in the AppCloud catalog.



Canvas apps (action, decision, and feeder services)

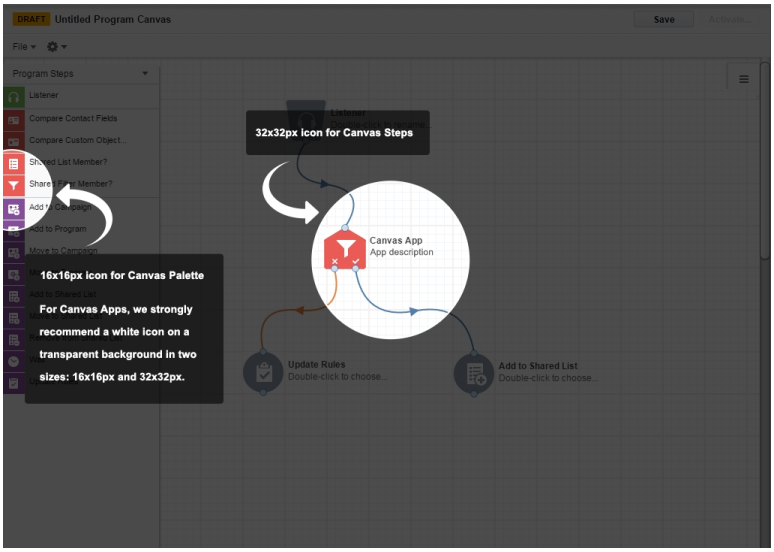
Size requirement:

- 32x32px
- 16x16px



Canvas apps are accessible from the campaign canvas and the program canvas left canvas palette. Two icons are needed for canvas apps:

- The 32x32px icon appears in the left campaign/program step on the canvas
- The 16x16px icon appears in the left canvas palette



Recommendation:

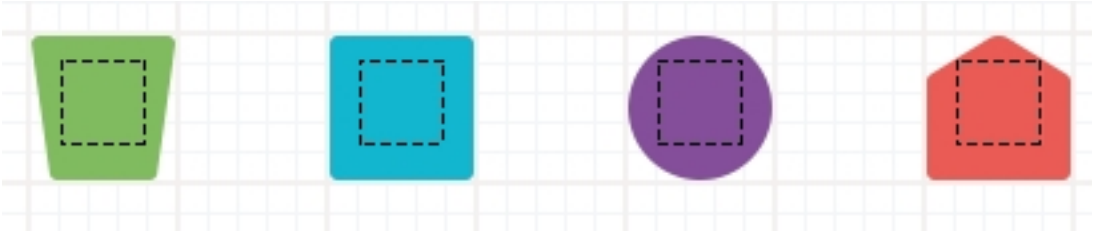
- White icon with transparent background

> **Important**: Because color is used to differentiate between audience (green), decision (red) and action (purple) steps and unpopulated state steps (gray), it is strongly recommended that app providers use white icons with transparent backgrounds for canvas apps.

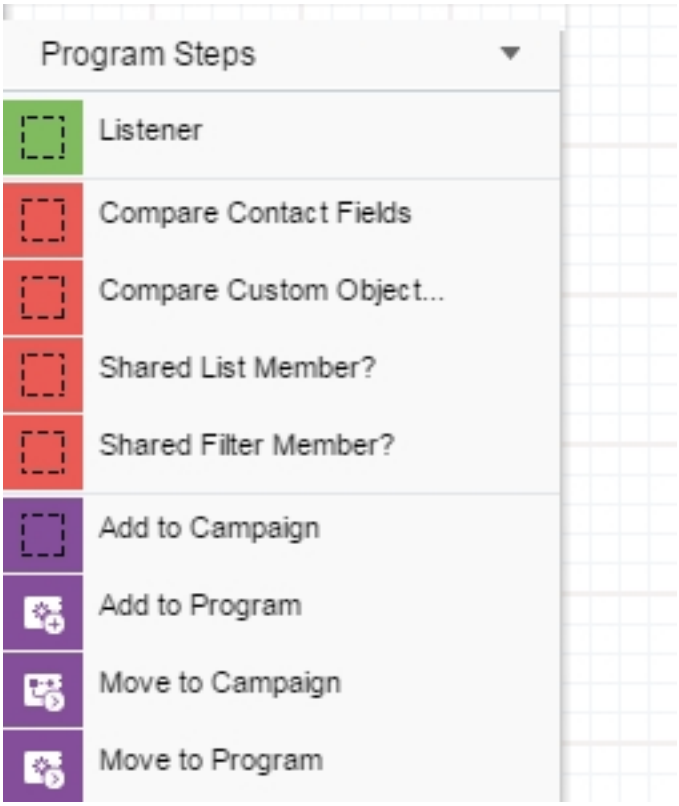
Migrating existing apps to the new interface

In the new interface, your existing apps will be modified accordingly:

- **Menu apps**: Unchanged, but scale in size.
- **Content apps**: Unchanged, but scale in size.
- **Firehose apps**: Unchanged, but scale in size.
- **Canvas apps**: The dotted line shows placement of the icon within the step:



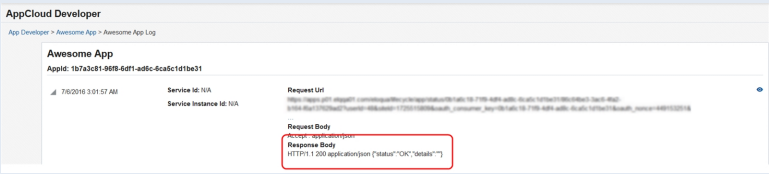
Left palette:



Viewing an app's outbound logs

The logs page records all the outbound requests made from Eloqua to your app's lifecycle URLs (the URLs an app calls when the app is configured, uninstalled, and so on). The records in the outbound log are useful for determining if Eloqua is calling your app's lifecycle URLs successfully.

Example: You can review the logs to determine if Eloqua is successfully calling the *Enable URL* by searching the logs page for the URL and viewing the response body to ensure a 200 level response code is returned.

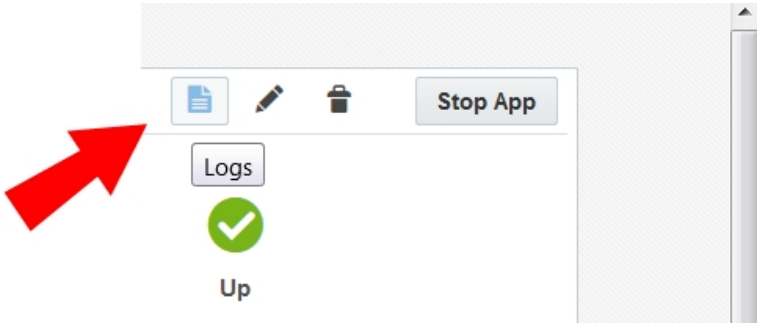


If Eloqua fails to call a URL, a 400 level response code is returned.

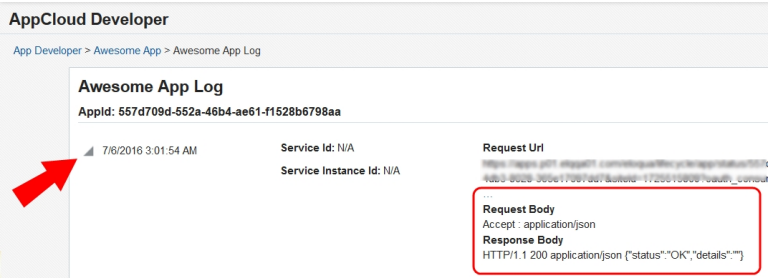
The logs page only displays outbound requests for your instance and other clients on your pod, to pull logs from another pod, use the app logs endpoint.

To view an app's outbound logs:

1. Navigate to **Settings** > **AppCloud Developer**.
2. Select your app and click **Logs**.

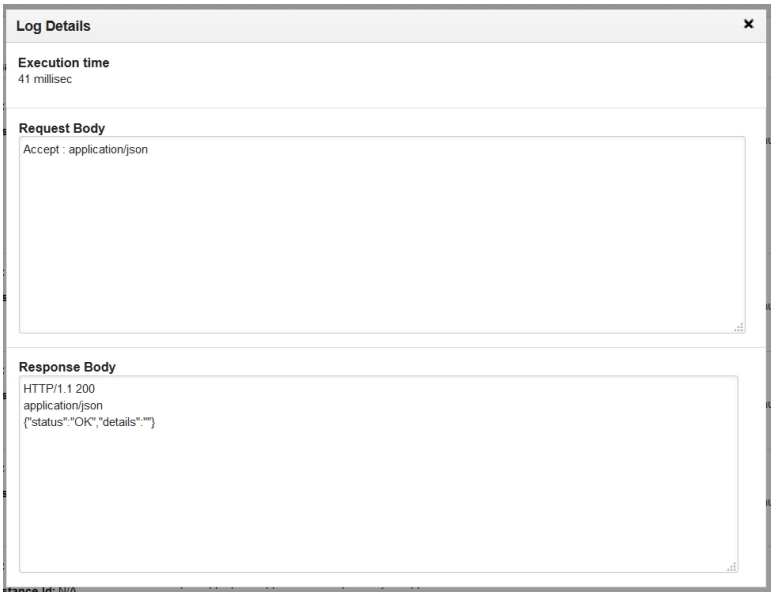


Expand the chevron to see the request and response body for the requests made to the lifecycle URLs.



Click **Log Details** to view details about the request.

The **Log Details** display how long the request took to execute, as well as the full request and response body.



Update or check your app's status using the cloud API

From time to time, you may wish to update your App's status. While it is possible to start or stop your app from the *App Details* page in the *AppCloud Developer*

Framework area of your Eloqua development instance, you can also do so programmatically using a cloud API endpoint. The endpoint also enables you to check the status of your app. You will need to know your app's ID in order to call out to the cloud API endpoints. The app ID is listed on the *App Details* page in your Eloqua development instance, and is a GUID.

The cloud API URL is `https://<eloqua base url>.com/api/cloud/1.0/<endpoint>`. All calls to the API must be authenticated using OAuth. Learn more about configuring and authenticating with OAuth.

Checking the app's status

Retrieve your app's status using the `GET /apps/{id}/configurations` endpoint:

```
GET /apps/a0f6779e-8263-4bfb-a292-9714cd10d1e7/configurations
```

The response should resemble:

```
HTTP/1.1 200 OK
{
   "uri": "/apps/a0f6779e-8263-4bfb-a292-9714cd10d1e7/configurations",
   "clientSecret":
"a1ba63f9aae1cc84d91544f15b0af27ab6ced4e42c6145b9cc0425742ecf2a0da1ba63f9aae
1cc84d91544f15b0af27ab6ce",
   "name": "Call Example",
   "description": "This is a test",
   "shortDescription": "This is a test",
   "iconUrl": "https://www.example.com/logo.png",
   "enableUrl": "http://example.com/enable/{appId}/{installId}/{userName}/
{siteName}/{siteId}",
   "statusUrl": "http://example.com/status/{installId}",
   "callbackUrl": "http://example.com/callback/{installId}",
   "makeAvailableUrl": "https://example.com",
   "publishedSites": [
```

```
      "AwesomeCompany"
   ],
   "status": "Up"
}
```

The `status` field indicates the app's current status, either "Up", "Down", or "Maintenance".

To update the app's status

Update the status by sending a request to the `PUT /apps/{id}/configurations` endpoint.

> **Important**: You must include all of the fields in the configuration document in the `PUT` request. You cannot merely specify a value for the `status` field.

The `status` can be "Up", "Down", or "Maintenance":

```
PUT /apps/a0f6779e-8263-4bfb-a292-9714cd10d1e7/configurations
{
   "uri": "/apps/a0f6779e-8263-4bfb-a292-9714cd10d1e7/configurations",
   "clientSecret":
"a1ba63f9aae1cc84d91544f15b0af27ab6ced4e42c6145b9cc0425742ecf2a0da1ba63f9aae
1cc84d91544f15b0af27ab6ce",
   "name": "Call Example",
   "description": "This is a test",
   "shortDescription": "This is a test",
   "iconUrl": "https://www.example.com/logo.png",
   "enableUrl": "http://example.com/enable/{appId}/{installId}/{userName}/
{siteName}/{siteId}",
   "statusUrl": "http://example.com/status/{installId}",
```

```
    "callbackUrl": "http://example.com/callback/{installId}",
    "makeAvailableUrl": "https://example.com",
    "publishedSites": [
       "AwesomeCompany"
    ],
    "status": "Maintenance"
}
```

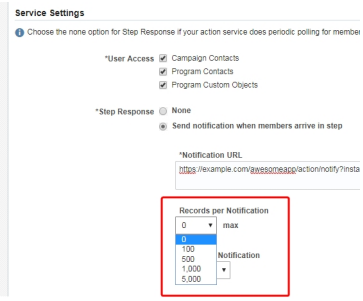
If successful, Eloqua will respond with an `HTTP/1.1 204 No Content`.

If you `GET` the `/apps/{id}/configurations` endpoint again, the response should reflect the updated status.

Retrieving app records using the bulk API

You can get notified when contacts or custom objects arrive to your app within a campaign or program. Eloqua will send your app a notification call in the form of a HTTP POST request that includes data about the contacts or custom object records. This is accomplished by supplying a **Notification URL** during service registration for action and decision services.

You control the maximum amount of contacts or custom object records sent per notification call by setting the **Records per Notification** option during service registration.



When **Records per Notification** is greater than 0, the HTTP POST request sent to the app will include data for the contacts or custom objects within the `items` array.

See an example

```
POST https://example.com/awesomeapp/action/notify?instance=
{InstanceId}&asset={AssetId}&execution={ExecutionId}

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 2,
   "count" : 2,
   "hasMore" : false,
   "items" :
   [
     {
       "ContactID" : "1",
       "EmailAddress" : "fred@example.com",
       "field1" : "stuff",
       "field2" : "things",
       "field3" : "et cetera"
     },
     {
       "ContactID" : "2",
       "EmailAddress" : "john@example.com",
       "field1" : "more stuff",
       "field2" : "other things",
       "field3" : "and so on"
     }
   ]
}
```

This request can get large, especially if the maximum records sent per notification is 5,000 for example. It is possible instead for app developers to select the maximum amount of **Records per Notification** to 0 and retrieve records on their own.

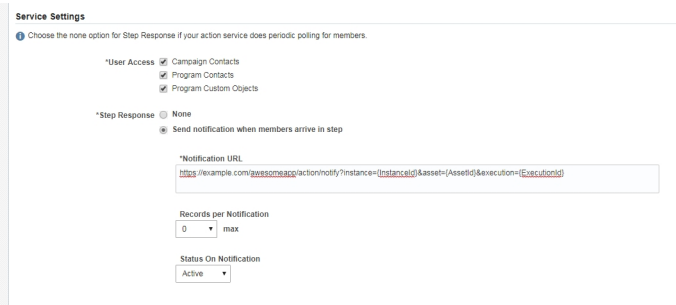
Setting records per notification to 0

If **Records per Notification** is set to 0, the notification call is sent with no data within the `items` property. The developer must then retrieve the records within the step using a separate bulk API export. This enables developers to control when and how quickly records are retrieved.

This tutorial will walkthrough the flow of having **Records per Notification** set to 0 and guiding developers through the flow of retrieving records within a step using a separate bulk API export.

Eloqua app service settings

Before we walkthrough how to respond to the notification call, note the Eloqua app service settings we will be using for the examples in this tutorial.



- Note the notification URL contains the 3 template parameters: `InstanceId` , `AssetId`, and `ExecutionId`.
- **Records per Notification** is set to **0**.

To edit your service settings, navigate to **Settings** > **AppCloud Developer** and edit your service's settings.

Notification call

When contacts or custom objects arrive in the step, Eloqua sends a HTTP POST request to the app's Notification URL.

```
POST https://example.com/awesomeapp/action/notify?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339&asset=456&execution=12345

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 0,
   "count" : 0,
   "hasMore" : false,
   "items" : []
}
```

Developers must then create a bulk API export definition using the data from Eloqua's call to the Notification URL.

Using contacts as an example, let's walkthrough how to form our bulk API export definition to retrieve the records.

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/exports

{
   "name": "Awesome App Contact Export - f82d50cd-86a9-4fca-b37e-4ec9a98b0339 - 12345",
   "fields": {
      "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
     "filter": "STATUS('{{ActionInstance (f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}') = 'pending'"
}
```

Response

```
{
   "name": "Awesome App Contact Export - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
   "fields": {
      "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "dataRetentionDuration": "PT12H",
   "uri": "/contacts/exports/55",
   "createdBy": "API.User",
   "createdAt": "2018-08-19T20:51:28.8201911Z",
   "updatedBy": "API.User",
   "updatedAt": "2018-08-19T20:51:28.8201911Z"
}
```

Next, create a sync using the export definition uri.

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
  "syncedInstanceUri" : "/contacts/exports/55",
  "callbackURL" : "https://www.example.com"
}
```

Response

```
{
   "syncedInstanceUri":"/contacts/exports/55",
   "callbackUrl":"http://www.example.com",
   "status":"pending",
   "createdAt":"2018-09-25T18:08:32.3485942Z",
   "createdBy":"API.User",
   "uri":"/syncs/66100"
}
```

The sync `status` will progress. Retrieve the sync to check the status of the sync.

Request

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/66100
```

Response


```
{
  "syncedInstanceURI": "/contacts/exports/55",
  "syncStartedAt": "2014-01-01T13:59:07.1375620Z",
  "syncEndedAt": "2014-01-01T13:59:14.1375620Z",
  "status": "success",
  "createdAt": "2014-01-01T13:59:07.1375620Z",
  "createdBy": "DocsExample",
  "uri": "/syncs/6"
}
```


After the sync has completed and the status is `success`, the last step is to retrieve the sync data.

Request

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/syncs/66100/data
```

Response


```
{
  "totalResults": 1,
  "limit": 0,
  "offset": 0,
  "count": 0,
  "hasMore": true,
  "items": [
    {
      "email": "john.snow@example.com"
```

```
  }
 ]
}
```

Using the data

Your app will likely want to use the records for a specific purpose, after this is done, you can import the data into Eloqua.

Importing the data into Eloqua

Start by creating a bulk import definition, setting the `status` to `complete` to import data. For this example, we will demonstrate using a contact import definition, where our service is an action service and its instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`.

If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a contact sync action definition.

> ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

Create a contact import definition.

Request

```
POST https: //secure.p03.eloqua.com/api/bulk/2.0/contacts/imports
{
```

```
 "name": "Awesome App Contact Import - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
 "updateRule": "always",
 "fields": {
  "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "syncActions": [
  {
   "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
   "action": "setStatus",
   "status": "complete"
  }
 ],
 "identifierFieldName": "emailAddress"
}
```

Eloqua's response will be a 201 Created response that includes a `uri` parameter, which you can use to identify the import.

Response

```
{
 "name": "Awesome App Contact Import - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
 "updateRule": "always",
 "fields": {
  "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "identifierFieldName": "emailAddress",
 "syncActions": [
  {
   "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
   "action": "setStatus",
   "status": "complete"
  }
 ],
```

```
  "isSyncTriggeredOnImport": false,
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/contacts/imports/6",
  "createdBy": "API.User",
  "createdAt": "2018-03-06T13:59:00.6600046Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-03-06T13:59:00.6600046Z"
}
```

Send Eloqua the import data using the `uri`:

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/6/data
[
  {
    "emailAddress": "john.snow@example.com"
  }
]
```

Eloqua's response will be a 204 No Content indicating that the data upload was successful.

App developer reference

The following pages provide detailed reference for the parameters and endpoints used in the Oracle Eloqua app Developer Framework.

- App level URL template parameters
- Service level URL template parameters
- App developer API endpoints
- Bulk API v2.0 documentation

❑ **Warning**: For `PUT` or `POST` requests, you must specify `application/json` in the **Content-Type** header. If the **Content-Type** header is not included, data persisted in Eloqua will be corrupted, resulting in your App not functioning correctly.

The **Content-Type** header is not required for `GET` or `DELETE` requests, since these do not accept data inputs.

Service level URL template parameters

The Oracle Eloqua app Developer Framework supports *URL Templating*, enabling the configuration of service URIs Eloqua calls out to. Any acceptable *template parameter* is substituted with the appropriate value before the call is made. See the Introduction to URL templating for more information.

There are different *template parameters* available for different calls in different AppCloud service types:

- Action service parameters
- Decision service parameters
- Feeder service parameters
- Content service parameters
- Menu service parameters
- Firehose service parameters

To look up individual parameters, please see the below parameter descriptions.

Action services

URL	Available parameters	Conditional Parameters
Create URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	
Configure URL	{InstanceId}, {InstallId}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	{AssetId} and {AssetName} are available **only if** the campaign has been saved with the referencing asset.
Copy URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, {OriginalAssetName}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	
Notification URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {ExecutionId}, {EntityType}, {CustomObjectId}	
Delete URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	

Decision services

URL	Available parameters	Conditional Parameters
Create URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	
Configure URL	{InstanceId}, {InstallId}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	{AssetId} and { AssetName } are available **only if** the campaign has been saved with the referencing asset.
Copy URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, {OriginalAssetName}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	
Notification URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {ExecutionId}, {EntityType}, {CustomObjectId}	
Delete URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	

Feeder services

URL	Available parameters	Conditional Parameters
Create URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	
Configure URL	{InstanceId}, {InstallId}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	{AssetId} and {AssetName} are available **only if** the campaign has been saved with the referencing asset.
Copy URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, {OriginalAssetName}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	
Notification URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {EventType}, {SiteId}, {AppId}, {EntityType}, {CustomObjectId}	
Delete URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	

Content services

URL	Available parameters	Conditional Parameters
Create URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	
Configure URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	{AssetId}, {AssetType} and {AssetName} are available **only if** the email or landing page has been saved with the referencing asset.
Copy URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {OriginalInstanceId}, {OriginalInstallId}, {OriginalAssetId}, {OriginalAssetName}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	
Email Notification URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {UserName}, {UserId}, {SiteName}, {SiteId}, {AppId}, {RenderType}, {IsPreview}, {ExecutionId}	
Landing Page Notification URL	{InstanceId}, {InstallId},{AssetId}, {AssetName}, {AssetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId} ,{VisitorId}, {CampaignId}	
Delete URL	{InstanceId}, {InstallId}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}	

Menu services

URL	Available parameters
Action URL	{InstallId}, {AssetId}, {AssetName}, {AssetType}, {AssetPath}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}

Firehose services

URL	Available parameters
Notification URL	{InstallId},{AssetId}, {AssetName}, {assetType}, {UserName}, {UserId}, {UserCulture}, {SiteName}, {SiteId}, {AppId}, {EventType}

Parameter descriptions

- `InstanceId` is a templated parameter used in AppCloud service apps. Its value is the GUID for the specific instance of the AppCloud service being used. For instance, when a user drags

and drops an action or decision service onto a campaign canvas this is considered a distinct instance and a new `InstanceId` is created.

- `InstallId` is a templated parameter used in AppCloud service apps. Its value is the GUID for the user's installation of the AppCloud App. Whenever a user installs an app, a new `InstallId` is created.

- `AssetId` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's ID. Each instance of an asset will have a unique `AssetId`.

- `AssetName` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's filename.

- `AssetType` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's type. Possible asset types include campaign, form, landing page, program, etc.

- `AssetPath` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's directory path.

- `OriginalInstanceId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `InstanceId` (as opposed to the copied service's `InstanceId`).

- `OriginalInstallId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `installId` (as opposed to the copied service's `InstallId`).

- `OriginalAssetId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `AssetId` (as opposed to the copied service's `AssetId`).

- `OriginalAssetName` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `AssetName` (as opposed to the copied service's `AssetName`).

- `UserName` is a templated parameter used in AppCloud service apps. Its value is the name of the user who triggered the call.

- `UserId` is a templated parameter used in AppCloud service apps. Its value is the ID of the user who triggered the call.

- `UserCulture` is a templated parameter used in AppCloud service apps. Its value is the linguistic profile of the user of the user that triggered the call. Eloqua uses Microsoft's CultureInfo class to define culture. For example, for English (United States), the code is "en-US", for French (Canada) it is "fr-CA".

- `SiteName` is a templated parameter used in AppCloud service apps. Its value is the name of the user's Eloqua instance.

- `SiteId` is a templated parameter used in AppCloud service apps. Its value is the GUID-based ID for the user's Eloqua instance.

- `AppId` is a templated parameter used in AppCloud service apps. Its value is the GUID-based ID of the app making the service call. Each AppCloud app has a single unique `AppId`.

- `EventType` is a templated parameter used in AppCloud service apps. Its value is the status change that triggered the call (Created, Activated, Draft, etc.)

  List of `EventType` values: `Created`, `Updated`, `Deleted`, `Activated`, `ActivatedBySchedule`, `Scheduled`, `Draft`, `DraftWaitingApproval`, `DraftApproved`, `CampaignAssetAdded`, `CampaignAssetRemoved`, `CampaignLandingPageAdded`, `CampaignLandingPageRemoved`, `CampaignFormAdded`, `CampaignFormRemoved`, `CampaignEmailAdded`, `CampaignEmailRemoved`, `CampaignSegmentAdded`, `CampaignSegmentRemoved`, `MembersAdded`, `ContentRequired`, `Completed`.

- `VisitorId` is a templated parameter used in AppCloud content service apps. Its value is an integer and represents the ID of the visitor for whom you wish to construct a landing page.

- `RenderType` is a templated parameter used in AppCloud content service apps. Its value specifies the different types of email renderings. Possible values are:
  - `0`: No render -- no content is being assembled.
  - `1`: EmailSend -- content is being assembled for sending an email.
  - `2`: LiveWeb -- content is being assembled to render a live landing page.
  - `3`: EmailPreview -- content is being assembled to render an email preview.
  - `4`: WebPreview -- content is being assembled to render a preview of a landing page.
  - `5`: EmailSaved -- content is being assembled to render the web version of a sent email.

- `IsPreview` is a templated parameter used in AppCloud content service apps. Its value specifies whether the referencing email is in preview mode. Possible values are:
  - `0`: False
  - `1`: True

- `ExecutionId` is a templated parameter used in AppCloud content service apps. Its value is an integer and identifies a unique email send execution.

- `EntityType` is a templated parameter used in AppCloud service apps. The value specifies the entity type the campaign or program is set to. Possible values are `Contacts` or `CustomObjectRecords`.

  For instance, if a user drags a decision step onto a custom object program, the `EntityType` would be `CustomObjectRecords` because the program's entity type is set to custom objects. Alternatively, if that decision step was dragged onto a contact program, the `EntityType` would be `Contacts`.

  Note that campaigns only support the contact entity type, but programs can support contacts or custom objects. Use the AssetType parameter to determine if the asset is a campaign or a program.

- `CustomObjectId` is a templated parameter used in AppCloud action, decision, and feeder service apps for programs. Its value is the ID of the custom object associated with the program when `CustomObjectRecords` is specified for EntityType. It must be an integer.

- `CampaignId` is a templated parameter used in Landing Page Content services. Its value is the ID of the campaign retrieved using the `elqCampaignId` query parameter. It must be an integer. Learn more.

App level URL template parameters

The Oracle Eloqua appDeveloper Framework supports *URL Templating*, enabling the configuration of service URIs Eloqua calls out to. Any acceptable *template parameter* is

substituted with the appropriate value before the call is made. See the Introduction to URL Templating page for more information.

To look up individual parameters, please see the below parameter descriptions.

App level parameters

Lifecycle parameters

URL	Description	Parameters
Enable URL	The URL called when the App is first installed or when it is reconfigured. This is a templated URL which should include, at a minimum, the {InstallId}, {AppId}, and {CallbackUrl} parameters. WIthout the callback URL you will not be able to complete the installation flow if your app requires configuring (for example, to use OAuth).	{InstallId}, {UserName}, {UserId}, UserCulture}, {SiteName}, {SiteId}, {AppId}, {CallbackUrl}
Configure URL	The URL called when clicking Configure (Gear icon next to Check Connection) for the App within the AppCloud Catalog.	{InstallId}, {UserName}, {UserId}, UserCulture}, {SiteName}, {SiteId}, {AppId}
Status URL	The URL called when a user checks the App's connection. Learn how to respond when Eloqua calls the Status URL.	{InstallId}, {UserName}, {UserId}, UserCulture}, {SiteName}, {SiteId}, {AppId}
Uninstall URL	The URL called when a user uninstalls the app. This allows Eloqua to notify the app provider when someone uninstalls their app from the appcloud catalog. This URL supports templated parameters such as {InstallId}, {AppId}, etc.	{InstallId}, {UserName}, {UserId}, UserCulture}, {SiteName}, {SiteId}, {AppId}

OAuth parameters

URL	Description	Parameters
Callback URL	Your App's registered redirection endpoint. Its	{InstallId}, {UserName}, {UserId}, UserCulture},

URL	Description	Parameters
	value should be the same as the `redirect_uri` used when requesting initial authorization for OAuth. Learn more about authentication with OAuth.	{SiteName}, {SiteId}, {AppId}

Parameter descriptions

- `InstanceId` is a templated parameter used in AppCloud service apps. Its value is the GUID for the specific instance of the AppCloud service being used. For instance, when a user drags and drops an action or decision service onto a campaign canvas this is considered a distinct instance and a new `InstanceId` is created.

- `InstallId` is a templated parameter used in AppCloud service apps. Its value is the GUID for the user's installation of the AppCloud App. Whenever a user installs an app, a new `InstallId` is created.

- `AssetId` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's ID. Each instance of an asset will have a unique `AssetId`.

- `AssetName` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's filename.

- `AssetType` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's type. Possible asset types include campaign, form, landing page, program, etc.

- `AssetPath` is a templated parameter used in AppCloud service apps. Its value is the referencing asset's directory path.

- `CallbackURL` is the Eloqua URL an app calls into when configuration is complete during installation.**Example**:<base url>/Apps/Cloud/Admin/Install/Callback/{1}?guid={2} Where `{1}` is the installId and `{2}` is a guid generated by Eloqua. **Note**: this parameter is separate and distinct from the OAuth Callback URL.

- `OriginalInstanceId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `InstanceId` (as opposed to the copied service's `InstanceId`).

- `OriginalInstallId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `installId` (as opposed to the copied service's `InstallId`).

- `OriginalAssetId` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `AssetId` (as opposed to the copied service's `AssetId`).

- `OriginalAssetName` is a templated parameter used in AppCloud service apps. It is specific to a service's Copy URL and its value is the original service's `AssetName` (as opposed to the copied service's `AssetName`).

- `UserName` is a templated parameter used in AppCloud service apps. Its value is the name of the user who triggered the call.

- `UserId` is a templated parameter used in AppCloud service apps. Its value is the ID of the user who triggered the call.

- `UserCulture` is a templated parameter used in AppCloud service apps. Its value is the linguistic profile of the user of the user that triggered the call. Eloqua uses Microsoft's CultureInfo class to define culture. For example, for English (United States), the code is "en-US", for French (Canada) it is "fr-CA".

- `SiteName` is a templated parameter used in AppCloud service apps. Its value is the name of the user's Eloqua instance.

- `SiteId` is a templated parameter used in AppCloud service apps. Its value is the GUID-based ID for the user's Eloqua instance.

- `AppId` is a templated parameter used in AppCloud service apps. Its value is the GUID-based ID of the app making the service call. Each AppCloud app has a single unique `AppId`.

- `EventType` is a templated parameter used in AppCloud service apps. Its value is the status change that triggered the call (Created, Activated, Draft, etc.)

  List of `EventType` values: `Created`, `Updated`, `Deleted`, `Activated`, `ActivatedBySchedule`, `Scheduled`, `Draft`, `DraftWaitingApproval`, `DraftApproved`, `CampaignAssetAdded`, `CampaignAssetRemoved`, `CampaignLandingPageAdded`, `CampaignLandingPageRemoved`, `CampaignFormAdded`, `CampaignFormRemoved`, `CampaignEmailAdded`, `CampaignEmailRemoved`, `CampaignSegmentAdded`, `CampaignSegmentRemoved`, `MembersAdded`, `ContentRequired`, `Completed`.

- `VisitorId` is a templated parameter used in AppCloud content service apps. Its value is an integer and represents the ID of the visitor for whom you wish to construct a landing page.

- `RenderType` is a templated parameter used in AppCloud content service apps. Its value specifies the different types of email renderings. Possible values are:
  - `0`: No render -- no content is being assembled.
  - `1`: EmailSend -- content is being assembled for sending an email.
  - `2`: LiveWeb -- content is being assembled to render a live landing page.
  - `3`: EmailPreview -- content is being assembled to render an email preview.
  - `4`: WebPreview -- content is being assembled to render a preview of a landing page.
  - `5`: EmailSaved -- content is being assembled to render the web version of a sent email.

- `IsPreview` is a templated parameter used in AppCloud content service apps. Its value specifies whether the referencing email is in preview mode. Possible values are:
  - `0`: False
  - `1`: True

- `ExecutionId` is a templated parameter used in AppCloud content service apps. Its value is an integer and identifies a unique email send execution.

Eloqua app developer API endpoints

Cloud API endpoints

- PUT /api/cloud/1.0/actions/instances/{id}
- PUT /api/cloud/1.0/contents/instances/{id}
- PUT /api/cloud/1.0/decisions/instances/{id}
- PUT /api/cloud/1.0/feeders/instances/{id}
- POST /api/cloud/1.0/actions/instances
- POST /api/cloud/1.0/decisions/instances

- POST /api/cloud/1.0/feeders/instances
- GET /api/cloud/1.0/apps/{id}/configurations
- PUT /api/cloud/1.0/apps/{id}/configurations
- GET /api/cloud/1.0/apps/{id}/logs

Learn more

PUT `/api/cloud/1.0/actions/instances/{id}`

Updates an action service instance.

❑ **Warning**: For `PUT` requests, you must specify `application/json` in the **Content-Type** header. If the **Content-Type** header is not included, data persisted in Eloqua will be corrupted, resulting in your App not functioning correctly.

URL parameters

name	type	description
`id`	GUID	Unique identifier for that service instance

Request Body

name	type	description
`requiresConfiguration`	boolean	`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be

name	type	description
		used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.
`recordDefinition`	Dictionary of strings	Defines the mapping between your fields and Eloqua's fields

Example

Update the record definition and configuration setting of the action service instance with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`:

```
PUT /api/cloud/1.0/actions/instances/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6
{
   "recordDefinition": {
      "ContactID": "{{Contact.Id}}",
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "requiresConfiguration": true
}
```

PUT `/api/cloud/1.0/contents/instances/{id}`

Updates a content service instance.

❑ **Warning**: For `PUT` requests, you must specify `application/json` in the **Content-Type** header. If the **Content-Type** header is not included, data persisted in Eloqua will be corrupted, resulting in your App not functioning correctly.

URL parameters

name	type	description
`id`	GUID	Unique identifier for that service instance

Request body

name	type	description
`requiresConfiguration`	boolean	`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.
`recordDefinition`	Dictionary of strings	Defines the mapping between your fields and Eloqua's fields
`height`	Integer	Height of the image
`width`	Integer	Width of the image
`editorImageUrl`	URL	URL of the image to display in the Landing Page and Email editor

Example

Update the record definition of the content instance with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`:

```
PUT /api/cloud/1.0/contents/instances/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6
{
  "recordDefinition": {
    "ContactID": "{{Contact.Id}}",
```

```
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
  },
  "height": 256,
  "width": 256,
  "editorImageUrl": "https://example.com/image.png"
}
```

PUT `/api/cloud/1.0/decisions/instances/{id}`

Updates a decision service instance.

> ❑ **Warning**: For `PUT` requests, you must specify `application/json` in the **Content-Type** header. If the **Content-Type** header is not included, data persisted in Eloqua will be corrupted, resulting in your App not functioning correctly.

URL parameters

name	type	description
`id`	GUID	Unique identifier for that service instance

Request Body

name	type	description
`requiresConfiguration`	boolean	`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to

name	type	description
		activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.
`recordDefinition`	Dictionary of strings	Defines the mapping between your fields and Eloqua's fields

Example

Update the record definition and configuration setting of the decision instance with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`:

```
PUT /api/cloud/1.0/decisions/instances/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6

{
   "recordDefinition" :
     {
       "ContactID" : "{{Contact.Id}}",
       "EmailAddress" : "{{Contact.Field(C_EmailAddress)}}"
     }
    "requiresConfiguration": true
}
```

PUT `/api/cloud/1.0/feeders/instances/{id}`

Updates a feeder service instance.

> ❑ **Warning**: For `PUT` requests, you must specify `application/json` in the **Content-Type** header. If the **Content-Type** header is not included, data persisted in Eloqua will be corrupted, resulting in your App not functioning correctly.

URL parameters

name	type	description
`id`	GUID	Unique identifier for that service instance

Request Body

name	type	description
`requiresConfiguration`	boolean	`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.

Example

Update the feeder service instance with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`:

```
PUT /api/cloud/1.0/feeders/instances/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6

{
  "requiresConfiguration": true
}
```

POST `/api/cloud/1.0/actions/instances`

Retrieves information for up to 200 action service instances, searched by service instance GUID.

Important: Request header 'X-HTTP-Method-Override' needs to be set with value 'SEARCH'.

Request parameters

Name	Type	Description	Possible values
ids	array of GUIDs	The unique identifiers for the service instances to retrieve. Maximum of 200 GUIDs per request. • If a GUID does not exist, or has been deleted, the result will be excluded from the response. • If the array contains duplicate GUIDs, or exceeds the 200 limit, a 400 Bad Request will be returned. • The dashes within the GUID are required.	d7a34c94-c370-4958-8a88-b56d5621e68a

Response parameters

Name	Type	Description
recordDefinition	Dictionary of strings	The field mapping used in the service instance.
requiresConfiguration	boolean	Whether or not user configuration is required before the

Name	Type	Description
		service can be used.
statement	string	The markup language statement used for referencing.
dependentAssetType	string	The asset type for which the service instance is being used. Possible values include: `Campaign` or `Program`. For example, if the service is on a campaign canvas, the `dependentAssetType` would be `Campaign`.
dependentAssetId	string	The asset Id for the asset for which the service instance is being used in.
uri	string	System-generated unique resource identifier.
createdBy	string	The login id of the user who created the service instance.
createdAt	string	The date and time the service instance was created, expressed in Unix time.
updatedBy	string	The login id of the user that last updated the service instance.
updatedAt	string	The date and time the service instance was last updated, expressed in Unix time.

Example

Retrieve 2 action service instances:

```
POST api/cloud/1.0/actions/instances
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
 "ids": [
  "d7a34c94-c370-4958-8a88-b56d5621e68a",
  "1869b464-56f2-4334-923a-e631849d3cc8"
 ]
}
```

Response:

```
{
 "items": [
  {
   "recordDefinition": {
    "Id": "{{Contact.Id}}",
    "field2": "{{Contact.Field(C_FirstName)}}"
   },
   "requiresConfiguration": false,
   "statement": "{{ActionInstance(d7a34c94c37049588a88b56d5621e68a)}}",
   "dependentAssetType": "Campaign",
   "dependentAssetId": 49251,
   "uri": "/actions/618068/instances/d7a34c94c37049588a88b56d5621e68a",
   "createdBy": "Admin.User",
   "createdAt": "2017-03-03T15:31:32.8930000Z",
   "updatedBy": "Admin.User",
   "updatedAt": "2017-03-03T15:31:33.4000000Z"
  },
  {
   "recordDefinition": {
```

```
      "Email": "{{CustomObject[1].Field[4]}}"
    },
    "requiresConfiguration": false,
    "statement": "{{ActionInstance(1869b46456f24334923ae631849d3cc8)}}",
    "dependentAssetType": "Program",
    "dependentAssetId": 49252,
    "uri": "/actions/618068/instances/1869b46456f24334923ae631849d3cc8",
    "createdBy": "Admin.User",
    "createdAt": "2017-03-03T15:31:33.8930000Z",
    "updatedBy": "Admin.User",
    "updatedAt": "2017-03-03T15:31:34.2170000Z"
   }
  ]
}
```


POST `/api/cloud/1.0/decisions/instances`

Retrieves information for up to 200 decision service instances, searched by service instance GUID.

> **Important**: Request header 'X-HTTP-Method-Override' needs to be set with value 'SEARCH'.

Request parameters

Name	Type	Description	Possible values
ids	array of GUIDs	The unique identifiers for the service instances to retrieve. Maximum of 200 GUIDs per request. • If a GUID does not exist, or has been	c8b0a315-4bb1-412f-b6d1-9c82f91cd6c2

Name	Type	Description	Possible values
		deleted, the result will be excluded from the response. • If the array contains duplicate GUIDs, or exceeds the 200 limit, a 400 Bad Request will be returned. • The dashes within the GUID are required.	

Response parameters

Name	Type	Description
recordDefinition	Dictionary of strings	The field mapping used in the service instance.
requiresConfiguration	boolean	Whether or not user configuration is required before the service can be used.
statement	string	The markup language statement used for referencing.
dependentAssetType	string	The asset type for which the service instance is being used. Possible values include: `Campaign` or `Program`. For example, if the service is on a campaign canvas, the `dependentAssetType` would be `Campaign`.

Name	Type	Description
dependentAssetId	string	The asset Id for the asset for which the service instance is being used in.
uri	string	System-generated unique resource identifier.
createdBy	string	The login id of the user who created the service instance.
createdAt	string	The date and time the service instance was created, expressed in Unix time.
updatedBy	string	The login id of the user that last updated the service instance.
updatedAt	string	The date and time the service instance was last updated, expressed in Unix time.

Example

Retrieve 2 decision service instances:

```
POST api/cloud/1.0/decisions/instances
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
  "ids": [
    "c8b0a315-4bb1-412f-b6d1-9c82f91cd6c2",
    "214e200f-7997-40fa-8329-5bdd847d7b60"
  ]
}
```

Important: Remember to include the dashes for the service instance GUIDs. A 400 validation error will be returned if the dashes are omitted.

Response:

```
{
  "items": [
    {
      "recordDefinition": {
        "Id": "{{Contact.Id}}",
        "field2": "{{Contact.Field(C_FirstName)}}"
      },
      "requiresConfiguration": false,
      "statement": "{{DecisionInstance(c8b0a3154bb1412fb6d19c82f91cd6c2)}}",
      "dependentAssetType": "Campaign",
      "dependentAssetId": 49254,
      "uri": "/decisions/618074/instances/c8b0a3154bb1412fb6d19c82f91cd6c2",
      "createdBy": "Admin.User",
      "createdAt": "2017-03-03T15:34:38.3470000Z",
      "updatedBy": "Admin.User",
      "updatedAt": "2017-03-03T15:34:38.9670000Z"
    },
    {
      "recordDefinition": {
        "Email": "{{CustomObject[1].Field[4]}}"
      },
      "requiresConfiguration": false,
```

```
    "statement": "{{DecisionInstance(214e200f799740fa83295bdd847d7b60)}}",
    "dependentAssetType": "Program",
    "dependentAssetId": 49255,
    "uri": "/decisions/618074/instances/214e200f799740fa83295bdd847d7b60",
    "createdBy": "Admin.User",
    "createdAt": "2017-03-03T15:34:39.3330000Z",
    "updatedBy": "Admin.User",
    "updatedAt": "2017-03-03T15:34:39.5930000Z"
   }
  ]
}
```


POST `/api/cloud/1.0/feeders/instances`

Retrieves information for up to 200 feeder service instances, searched by service instance GUID.

> **Important**: Request header 'X-HTTP-Method-Override' needs to be set with value 'SEARCH'.

Request parameters

Name	Type	Description	Possible values
ids	array of GUIDs	The unique identifiers for the service instances to retrieve. Maximum of 200 GUIDs per request. • If a GUID does not exist, or has been deleted, the result will be excluded from the response.	4de74ef1-bb2c-44d6-97ba-6acb1565bc58

Name	Type	Description	Possible values
		• If the array contains duplicate GUIDs, or exceeds the 200 limit, a 400 Bad Request will be returned. • The dashes within the GUID are required.	

Response parameters

Name	Type	Description
statement	string	The markup language statement used for referencing.
dependentAssetType	string	The asset type for which the service instance is being used. Possible values include: `Campaign` or `Program`. For example, if the service is on a campaign canvas, the `dependentAssetType` would be `Campaign`.
dependentAssetId	string	The asset Id for the asset for which the service instance is being used in.
requiresConfiguration	boolean	Whether or not user configuration is required before the service can be used.
uri	string	System-generated unique resource

Name	Type	Description
		identifier.
createdBy	string	The login id of the user who created the service instance.
createdAt	string	The date and time the service instance was created, expressed in Unix time.
updatedBy	string	The login id of the user that last updated the service instance.
updatedAt	string	The date and time the service instance was last updated, expressed in Unix time.

Example

Retrieve 2 feeder service instances:

```
POST api/cloud/1.0/feeders/instances
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
  "ids": [
    "4de74ef1-bb2c-44d6-97ba-6acb1565bc58",
    "97e3f90e-6ef8-40a8-9554-588a72de440e"
  ]
}
```

Important: Remember to include the dashes for the service instance GUIDs. A 400 validation error will be returned if the dashes are omitted.

Response:


```
{
  "items": [
    {
      "statement": "{{FeederInstance(4de74ef1bb2c44d697ba6acb1565bc58)}}",
      "dependentAssetType": "Campaign",
      "dependentAssetId": 6662,
      "requiresConfiguration": true,
      "uri": "/feeders/90289/instances/4de74ef1bb2c44d697ba6acb1565bc58",
      "createdBy": "API.User",
      "createdAt": "2018-06-01T17:57:00.1470000Z",
      "updatedBy": "API.User",
      "updatedAt": "2018-06-01T17:57:23.5130000Z"
    },
    {
      "statement": "{{FeederInstance(97e3f90e6ef840a89554588a72de440e)}}",
      "dependentAssetType": "Program",
      "dependentAssetId": 1385,
      "requiresConfiguration": true,
      "uri": "/feeders/90289/instances/97e3f90e6ef840a89554588a72de440e",
      "createdBy": "API.User",
      "createdAt": "2018-06-01T18:10:21.5700000Z",
      "updatedBy": "API.User",
      "updatedAt": "2018-06-01T18:10:36.8770000Z"
    }
  ]
}
```


GET `/api/cloud/1.0/apps/{id}/configurations`

Retrieves an Oracle Eloqua app App configuration.

URL parameters

name	type	description
`id`	GUID	The AppId of the app configuration you are retrieving

Request body

None.

Example

Example: Retrieve the App configuration file for the App with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`

```
GET /api/cloud/1.0/apps/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6/configurations
```

PUT `/api/cloud/1.0/apps/{id}/configurations`

Updates an Oracle Eloqua app App configuration.

Warning: changes to the configuration file directly affect the app. Accidentally changing the `enableUrl`, for example, will result in people being unable to install your app. Ensure that any changes you push out are intentional.

URL parameters

name	type	description
`id`	GUID	The AppId of the app configuration you are updating

Request Body

name	type	description
`requiresConfiguration`	boolean	`requiresConfiguration` is an optional Boolean parameter which tells Eloqua whether user configuration is required before the app can be used. If set to `true`, users will be unable to activate a campaign or program containing the unconfigured app service instance. Eloqua will display an error message.
`uri`	URI	specific URI that describes the app
`clientSecret`	token	Eloqua-generated OAuth token
`name`	string	name of the app
`description`	string	description of the app
`shortDescription`	string	a brief description of the app
`iconUrl`	URL	URL pointing to the icon that Eloqua should display for the app
`enableUrl`	templated URL	URL called when the App is first installed or when it is reconfigured
`statusUrl`	templated URL	URL called when a user checks the app's status
`callbackUrl`	templated URL	URL that the users should be redirected to after installing an app
`makeAvailableUrl`	URL	Eloqua-generated URL used to add the

name	type	description
		app to a user's catalogue
`publishedSites`	dictionary of strings	lists the sites where the app is installed
`status`	string	Describes the status of the App - either "Up", "Down", or "Maintenance".

Example

Example: Set the status for the App with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6` to **Down**.

```
PUT /api/cloud/1.0/apps/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6/configurations
{
   "uri": "/apps/a0f6779e-8263-4bfb-a292-9714cd10d1e7/configurations",
   "clientSecret":
"a1ba63f9aae1cc84d91544f15b0af27ab6ced4e42c6145b9cc0425742ecf2a0da1ba63f9
aae1cc84d91544f15b0af27ab6ce",
   "name": "Call Example",
   "description": "This is a test",
   "shortDescription": "This is a test",
   "iconUrl": "https://www.example.com/logo.png",
   "enableUrl": "http://example.com/enable/{appId}/{installId}/{userName}/
{siteName}/{siteId}",
   "statusUrl": "http://example.com/status/{installId}",
   "callbackUrl": "http://example.com/callback/{installId}",
   "makeAvailableUrl": "https://example.com",
   "publishedSites": [
      "AwesomeCompany"
   ],
   "status": "Down"
}
```

GET `/api/cloud/1.0/apps/{id}/logs`

Retrieves an app's outbound logs. This endpoint can be used to retrieve logs for apps across different pods, while the logs page in the Eloqua interface is limited only to displaying outbound requests for your instance and other clients on your pod.

This endpoint orders results by the request date, where the latest outbound requests are ordered first. The outbound logs returned in the request will differ depending on the authentication used to access the endpoint.

If authenticated using oAuth:

- The outbound logs for a provider's apps are displayed. Attempting to access logs for other apps will return a 403 forbidden response code.

If authenticated using basic authentication:

- You can access logs for each app within your instance.

URL parameters

name	type	description
`id`	GUID	The AppId of the app log you are retrieving.
`serviceInstanceId`	GUID	The service instance Id of the app you are retrieving.
`startDate`	dateTime (10 digit integer Unix time)	The start date to filter outbound logs.
`endDate`	dateTime (10 digit integer Unix time)	The end date to filter outbound logs.
`limit`	integer	A URL parameter that specifies the maximum number of records to return.
`offset`	integer	Specifies an offset that allows to

name	type	description
		retrieve the next batch of records.
`totalResults`	boolean	A URL parameter that captures the total number of records that satisfy the request. This is not the count returned in the current response, but total count on the server side.

Request Body

None.

Example

Example: Retrieve the outbound logs for the App configuration with GUID `7b95fe48-6598-43e8-8fd1-dcb40cf83ef6`

```
GET /api/cloud/1.0/apps/7b95fe48-6598-43e8-8fd1-dcb40cf83ef6/logs
```

App developer frequently asked questions

Why should I develop apps for the app developer framework?

The app developer framework is a complete development platform where you can easily build, register, and deploy apps for Eloqua. With new and improved service types which take advantage of Eloqua's bulk API, support for OAuth, and the ability to test your applications with Eloqua prior to launch, the App Developer Framework provides the environment needed to put apps first.

What permissions do I need to start developing apps for the Oracle Eloqua app?

At the minimum, you need access to an Eloqua instance. If you are not currently an Eloqua user, you can sign up as a technology partner to obtain a development instance. As a user, you will also need the **Advanced Users - Marketing** permissions.

When and how does my app get listed on the Oracle Eloqua app site?

See building apps for the Oracle marketing app developer framework on Topliners for detailed instructions.

How do marketers find my app to start using it?

Registered apps are listed on the Oracle Cloud Marketplace. Users are linked to this page through the "Get more Apps" link in Eloqua's **AppCloud Catalog** section.

No. The Oracle Eloqua app exclusively supports contact, account, activity and custom object fields in its record definition fields. You must use the correct Eloqua markup language statement to reference each field.

Is there an Oracle Eloqua app certification program?

Yes. Check out the Oracle Marketing AppCloud Certification Program on Topliners.

What kind of digital certificate is required to communicate with Eloqua?

App's URL endpoints must have an SSL Certificate, specifically needing to be a digital signature from a certificate authority (CA). Eloqua will not communicate with untrusted sites.

What happens during an Eloqua release?

The application and the various associated services will be intermittently unavailable within the duration of the maintenance window. If there are any Bulk API sync failures, they should be retried after the maintenance window completes.

Which version of TLS is supported?

TLS 1.2 is supported.

Oracle Eloqua App Services and Operations

Which service should I develop?

It depends on what you're trying to achieve! See the Service Descriptions for an overview of each service and the use cases it supports.

What's the difference between Oracle Eloqua App services and Cloud Connectors or Components?

Eloqua Oracle Eloqua app services greatly extend the functionality provided by cloud connectors and cloud components. Cloud content replaces cloud components, allowing you to process emails in bulk and test the service within Eloqua. Unlike cloud

connectors and components, the Oracle Eloqua App services use Eloqua's bulk API for processing, greatly improving performance and throughput.

Can I include campaign, email, landing page, form fields in my record definition?

No. The Oracle Eloqua app exclusively supports contact, account, activity and custom object fields in its record definition fields. You must use the correct Eloqua markup language statement to reference each field.

Can I include static values in my record definition?

No. You can only specify the Eloqua markup language for Eloqua fields.

Managing Your Apps

What does this app status mean?

See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

What if my request to refresh a token times out?

If the current access token has not been used, submitting a request to authenticate with the previous refresh token will return the existing new access token and refresh token.

What happens if contacts encounter an error in an Action or Decision step? What happens if the service is unavailable when contacts flow into a step on the canvas?

The contacts remain in the step until the marketer manually pushes them into the next step on the canvas.

Does Eloqua notify me when someone uninstalls or deletes my app from their Eloqua instance?

If you set an Uninstall URL it will be called when a user uninstalls the app.

Why has my app been shut down?

If in the last five minutes, there were more than 100 calls, and 25% of them failed, the app will be shut down. All the pending and future requests will be marked as failed. See the App shutdown for more information.

Will Eloqua retry my contacts if my app doesn't respond?

No, the contacts in a step will be marked as "Errored". If the marketer has configured a default path for the contacts to flow through, then the contacts will flow into the next step.

What if my app responds to the notify call after 100 seconds?

Any response after 100 seconds will be ignored.

What if my app responds to the notify call with a response status code that is not 200 level?

If Eloqua calls out to your app and receives a response status code that is between 300 and 599, Eloqua will retry the notify call over approximately an eight-hour period of time with a backoff strategy of the time between calls doubling after each call. After this eight-hour period, the contacts in a step will be marked as "Errored". If the marketer has configured a default path for the contacts to flow through, then the contacts will flow into the next step.

Why does my app prompt the error "502 Error: App Provider not Available"?

This error message will appear if a URL has been configured with a non-standard port. Eloqua only supports the standard ports 80 and 443. If a URL is not configured with port 80 or port 443, requests to this URL will fail.

Limits

Are there limits that I should be aware of?

The AppCloud developer framework relies on the bulk API. The bulk API has limits on the size of the staging area for imports and exports, on the amount of data you can import at one time, and on the number of fields you can include in a record definition. There is also a daily limit on the number of syncs you can perform.

What happens if I reach the daily sync limits?

The daily sync limit is not currently enforced, but syncs are logged and monitored.

Oracle Eloqua

Developer Guide

Contents

Application API

Oracle Eloqua's application API is a RESTful API that extends the functionality of Eloqua's automation engines, program builder, and campaign canvas, and can be used to build applications for Eloqua's AppCloud.

The application API is primarily used for asset management. Assets are the building blocks of Eloqua marketing campaigns, such as campaign elements (emails and landing pages) and components (images, file storage, hyperlinks, field merges, email headers and footers, shared and dynamic content sections, signature layouts, and rules).

Eloqua's application API is synchronous and so is not recommended for use with high volumes of data. Use Eloqua's bulk API for high volumes of data.

An application API call consists of an initial authentication and the following API call format:

```
[Verb] [base URL]/API/[APIName]/[APINumber]/[endpoint]
```

For example, a typical API call using the application API's 1.0 REST API would look something like:

```
GET [base URL]/API/REST/1.0/assets/emails
```

A typical API call using application API's 2.0 REST API would look something like:

```
GET [base URL]/API/REST/2.0/assets/campaigns
```

Learn more about API requests.

Tutorials

Detailed walkthroughs for developers on how to use the application API endpoints.

Reference

API reference

Activity detail values

Campaign element reference

Endpoints

Email deployment API

Signature rules API

Retrieve account information for up to 200 accounts

Retrieve contact information for up to 200 contacts

Email deployment API

The email deployment API is used to send and retrieve email deployments. There are three methods for creating and sending email deployments:

- To a single contact
- To a low volume of contacts (up to 100 contacts)

- To a large numbers of contacts (up to 2000 contacts)

API tasks

Create and send an email deployment to one contact

Retrieve an email deployment sent to one contact

Create and send an email deployment to a small number of contacts

Retrieve an email deployment sent to a small number of contacts

Create and send an email deployment to a large number of contacts

Retrieve an email deployment sent to a large number of contacts

POST `api/REST/2.0/assets/email/deployment`

Send an email to a single contact. This quick send endpoint is used to create and send an email deployment for an existing email asset to a single contact.

Request parameters

Required

Name	Type	Description	Possible values
type	string	The type of email deployment.	`EmailTestDeployment` **Note**: The activity's email send type will appear as

Name	Type	Description	Possible values
			`QuickSend`.
name	string	The name of the deployment	
contactId	string	The contact ID of the contact to send to.	

Nested schema: `email`

The email to send. The selected email cannot be modified as part of an email deployment request.

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset.	

> **Note**: The name is not actually read so any value can be used but it is still required.

Optional

Name	Type	Description	Possible values
sendFromUserId	string	The ID of the user to use as the sender of the email. The default will use the current user.	

Nested Schema: sendOptions

Name	Type	Description	Possible values
allowResend	string	Allow resending this email asset to the contact. The default will be true.	
allowSendToMasterExclude	string	Allow sending to contacts in the master exclude list. The default will be false.	
allowSendToUnsubscribe	string	Allow sending to globally unsubscribed contacts. The default will be false.	
allowSendToGroupUnsubscribe	string	Allow sending to contacts unsubscribed from the email group. The default will be true.	
allowSendToBounceback	string	Allow sending to contacts that	

Name	Type	Description	Possible values
		are marked as having bounced. The default will be false.	

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment	`EmailTestDeployment` **Note**: The activity's Email Send Type will appear as `QuickSend`.
name	string	The name of the deployment	
contactId	string	The contact ID of the contact to send to	
successfulSendCount	string	The number of emails which have been	

Name	Type	Description	Possible values
		successfully sent so far	
failedSendCount	string	The number of emails which failed during send so far	
currentStatus	string	Current deployment status	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
sendFromUserId	string	The ID of the user used as the sender of the email	
permissions	array	Permission values of the deployment granted to the current user	
depth	string	The request's level of detail	• `minmal` • `partial` • `complete`

Nested schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset	

Note: For a full list of parameters returned within the email object see Retrieve an email response parameters.

Nested Schema: sendOptions

Name	Type	Description	Possible values
type	string	The type of send options	`EmailSendOptions`
allowResend	string	Whether or not this email is allowed to be resent to the contact	
allowSendToMasterExclude	string	Whether or not this email is allowed to be sent to contacts in the master	

Name	Type	Description	Possible values
		exclude list	
allowSendToUnsubscribe	string	Whether or not this email is allowed to be sent to globally unsubscribed contacts	
allowSendToGroupUnsubscribe	string	Whether or not this email is allowed to be sent to contacts unsubscribed from the email group	
allowSendToBounceback	string	Whether or not this email is allowed to be sent to contacts that are marked as having bounced	

Example

Create and send an email deployment for an existing email asset to a single contact:

```
POST /assets/email/deployment
```

Request body:

```
{
  "type": "EmailTestDeployment",
```

```
  "name": "REST Test 01",
  "contactId":  "1",
  "email": {
    "type": "Email",
    "id": "100",
    "name": "test01"
  },
  "sendOptions": {
    "allowResend": "true",
    "allowSendToUnsubscribe": "false"
  }
}
```

Response:

```
{
  "type": "EmailTestDeployment",
  "currentStatus": "normal",
  "id": "4",
  "depth": "complete",
  "name": "REST Test 01",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update",
    "Activate"
  ],
  "email": {
    "type": "Email",
    "x_e10_previewUrl":
"https://p03.eloquapreview.com/Preview.aspx?siteId=238011564&userGuid=a2475311-
77bb-41e6-9db1-45c8c6402efa",
    "x_e10_previewExpiryDate": "1468620740",
    "x_e10_isTemplate": "false",
    "x_e10_createdAt": "1468620436",
    "x_e10_createdBy": "11",
    "currentStatus": "Draft",
    "id": "100",
```

```
  "createdAt": "1468620389",
  "createdBy": "11",
  "depth": "complete",
  "folderId": "42",
  "name": "Test_Email",
  "permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update"
  ],
  "updatedAt": "1468620436",
  "updatedBy": "11",
  "archive": "false",
  "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
  "contentSections": [],
  "dynamicContents": [],
  "emailFooterId": "1",
  "emailGroupId": "4",
  "emailHeaderId": "1",
  "encodingId": "3",
  "fieldMerges": [],
  "forms": [],
  "htmlContent": {
   "type": "RawHtmlContent",
   "contentSource": "upload",
   "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <body>\r\n
<p>Test Email</p>\r\n  </body> \r\n</html>"
  },
  "hyperlinks": [],
  "images": [],
  "isContentProtected": "false",
  "isPlainTextEditable": "false",
  "isPrivate": "False",
  "isTracked": "true",
  "layout": "{}",
  "plainText": "\r\nTest Email\r\n\r\n",
  "renderMode": "Flow",
  "replyToEmail": "API.User@test.oracle.com",
  "replyToName": "API User",
```

```
    "sendPlainTextOnly": "false",
    "senderEmail": "API.User@test.oracle.com",
    "senderName": "API User",
    "style": "{}",
    "subject": "Test"
  },
  "failedSendCount": "0",
  "successfulSendCount": "0",
  "contactId": "1",
  "sendOptions": {
    "type": "EmailSendOptions",
    "allowResend": "true",
    "allowSendToBounceback": "false",
    "allowSendToGroupUnsubscribe": "true",
    "allowSendToMasterExclude": "false",
    "allowSendToUnsubscribe": "false"
  }
}
```


GET `api/REST/2.0/assets/email/deployment/{id}`

Retrieve an email deployment sent to one contact.

Request parameters

Required

Name	Type	Description	Possible values
id	integer	Unique identifier of the email deployment.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will be complete. Learn more about the depth parameter.	• `minmal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment	`EmailTestDeployment` **Note**: The activity's Email Send Type will appear as `QuickSend`.
name	string	The name of the deployment	
contactId	string	The contact ID of the contact to send to	
successfulSendCount	string	The number of emails which have been	

Name	Type	Description	Possible values
		successfully sent so far	
failedSendCount	string	The number of emails which failed during send so far	
currentStatus	string	Current deployment status	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
sendFromUserId	string	The ID of the user used as the sender of the email	
permissions	array	Permission values of the deployment granted to the current user	
depth	string	The request's level of detail	• `minmal` • `partial` • `complete`

Nested schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset	

Note: For a full list of parameters returned within the email object see Retrieve an email response parameters.

Nested Schema: sendOptions

Name	Type	Description	Possible values
type	string	The type of send options	`EmailSendOptions`
allowResend	string	Whether or not this email is allowed to be resent to the contact	
allowSendToMasterExclude	string	Whether or not this email is allowed to be sent to contacts in the master	

Name	Type	Description	Possible values
		exclude list	
allowSendToUnsubscribe	string	Whether or not this email is allowed to be sent to globally unsubscribed contacts	
allowSendToGroupUnsubscribe	string	Whether or not this email is allowed to be sent to contacts unsubscribed from the email group	
allowSendToBounceback	string	Whether or not this email is allowed to be sent to contacts that are marked as having bounced	

Example

Retrieve an email deployment with the id = 4:

```
GET /assets/email/deployment/4
```

Response:

```
{
  "type": "EmailTestDeployment",
```

```
 "currentStatus": "normal",
 "id": "4",
 "depth": "complete",
 "name": "REST Test 01",
 "permissions": [
  "Retrieve",
  "SetSecurity",
  "Delete",
  "Update",
  "Activate"
 ],
 "email": {
  "type": "Email",
  "currentStatus": "Draft",
  "id": "100",
  "createdAt": "1468620389",
  "createdBy": "11",
  "depth": "complete",
  "folderId": "42",
  "name": "Test_Email",
  "permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update"
  ],
  "updatedAt": "1468620436",
  "updatedBy": "11",
  "archive": "false",
  "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
  "contentSections": [],
  "dynamicContents": [],
  "emailFooterId": "1",
  "emailGroupId": "4",
  "emailHeaderId": "1",
  "encodingId": "3",
  "fieldMerges": [],
  "forms": [],
  "htmlContent": {
   "type": "RawHtmlContent",
```

```
    "contentSource": "upload",
    "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <body>\r\n
<p>Test Email</p>\r\n  </body> \r\n</html>"
   },
   "hyperlinks": [],
   "images": [],
   "isContentProtected": "false",
   "isPlainTextEditable": "false",
   "isPrivate": "False",
   "isTracked": "true",
   "layout": "{}",
   "plainText": "\r\nTest Email\r\n\r\n",
   "renderMode": "Flow",
   "replyToEmail": "api.user@test.oracle.com",
   "replyToName": "API User",
   "sendPlainTextOnly": "false",
   "senderEmail": "api.user@test.oracle.com",
   "senderName": "API User",
   "style": "{}",
   "subject": "Test"
  },
  "endAt": "1468620440",
  "failedSendCount": "0",
  "successfulSendCount": "1",
  "contactId": "1",
  "sendOptions": {
   "type": "EmailSendOptions",
   "allowResend": "true",
   "allowSendToBounceback": "false",
   "allowSendToGroupUnsubscribe": "true",
   "allowSendToMasterExclude": "false",
   "allowSendToUnsubscribe": "false"
  }
}
```


POST `api/REST/2.0/assets/email/deployment`

This low volume endpoint is used to create and send an email deployment for an existing email asset to up to 100 contacts.

Request parameters

Required

Name	Type	Description	Possible values
type	string	The type of email deployment.	`EmailLowVolumeDeployment` **Note**: The activity's email send type will appear as `LowVolumeAPI`.
name	string	The name of the deployment.	
contactIds	array	The array of contact IDs of the contacts to send to. The number of contactIds in the array is limited to: Minimum: 1, Maximum: 100.	

Nested schema: `email`

The email to send. The selected email cannot be modified as part of an email deployment request.

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset.	

Name	Type	Description	Possible values
		Note: The name is not actually read so any value can be used but it is still required.	

Optional

Name	Type	Description	Possible values
sendFromUserId	string	The ID of the user to use as the sender of the email. If specified, then the `signatureRuleId` must not be specified. The default will use the current user.	
signatureRuleId	string	The asset id of the signature rule to use. If specified then the `sendFromUserId` must not be specified.	
notificationEmailAddress	string	The email address of the user to send a deployment notification email to. Must be the email address in the user's profile, in email address format.	
sendDate	string	A date in the future to schedule the deployment. If the date is in the past, or not specified, the deployment is scheduled immediately. The format is Unix Time.	

Nested schema: sendOptions

Name	Type	Description	Possible values
allowResend	string	Allow resending this email asset to the contact. The default is true.	
allowSendToMasterExclude	string	Allow sending to contacts in the master exclude list. The default is false.	
allowSendToUnsubscribe	string	Allow sending to globally unsubscribed contacts. The default is false.	
allowSendToGroupUnsubscribe	string	Allow sending to contacts unsubscribed from the email group. The default is true.	
allowSendToBounceback	string	Allow sending to contacts that are marked as having bounced. The default is false.	

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment.	`EmailLowVolumeDeployment` **Note**: The activity's Email Send Type will appear as `LowVolumeAPI`.
name	string	The name of the deployment.	
contactIds	array	The array of contact ids of the contacts to the	

Name	Type	Description	Possible values
		deployment will attempt to send to. Contacts that were filtered out during deployment creation because of `sendOptions` are removed from this array and are included in the `exclusions` array.	
exclusions	array	The array of errors describing the reason each contact was excluded from the deployment.	
successfulSendCount	string	The number of emails which have been successfully sent so far.	
failedSendCount	string	The number of emails which failed during send so far.	
currentStatus	string	Current deployment status.	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
sendFromUserId	string	The ID of the user used as the sender of the email.	

Name	Type	Description	Possible values
		Note: If `signatureRuleId` is set, `sendFromUserId` will display the user id of the user who sent the request.	
signatureRuleId	string	The ID of the signature rule used if specified on creation.	
sendDate	string	The date the deployment was scheduled for send. The format is Unix Time.	
notificationEmailAddress	string	The email address of the user which was sent a deployment notification email if specified on creation.	
permissions	array	Permission values of the deployment granted to the current user.	
depth	string	The request's level of detail.	`complete`

Nested schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`Email`
id	string	The ID of the email asset.	
name	string	The name of the email asset.	

Note: For a full list of parameters returned within the email object see Retrieve an email response parameters.

Nested schema: sendOptions

Name	Type	Description	Possible values
type	string	The type of send options	`EmailSendOptions`
allowResend	string	Whether or not this email is allowed to be resent to the contact.	
allowSendToMasterExclude	string	Whether or not this email is allowed to be sent to contacts in the master	

Name	Type	Description	Possible values
		exclude list.	
allowSendToUnsubscribe	string	Whether or not this email is allowed to be sent to globally unsubscribed contacts.	
allowSendToGroupUnsubscribe	string	Whether or not this email is allowed to be sent to contacts unsubscribed from the email group.	
allowSendToBounceback	string	Whether or not this email is allowed to be sent to contacts that are marked as having bounced.	

Example

Create and send an email deployment for an existing email asset to two contacts:

```
POST /assets/email/deployment
```

Request body:

```
{
                                        "type": "EmailLowVolumeDeployment",
```

```
                "name": "REST Low Volume 01",
                "contactIds": [
                "1",
                "2"
                ],
                "email": {
                "type": "Email",
                "id": "101",
                "name": "test01"
                },
                "signatureRuleId": "1",
                "notificationEmailAddress": "api.user@test.oracle.com",
                "sendDate": "1468622700",
                "sendOptions": {
                "allowResend": "true",
                "allowSendToUnsubscribe": "false"
                }
        }
```


Response:


```
{
                "type": "EmailLowVolumeDeployment",
                "currentStatus": "normal",
                "id": "5",
                "depth": "complete",
                "name": "REST Low Volume 01",
                "permissions": [
                "Retrieve",
                "SetSecurity",
                "Delete",
                "Update",
                "Activate"
                ],
                "email": {
                "type": "Email",
                "currentStatus": "Draft",
                "id": "101",
                "createdAt": "1468621285",
```

```
                                    "createdBy": "11",
                                    "depth": "complete",
                                    "folderId": "42",
                                    "name": "Test_Low_Volume_Email",
                                    "permissions": [
                                    "Retrieve",
                                    "SetSecurity",
                                    "Delete",
                                    "Update"
                                    ],
                                    "updatedAt": "1468621363",
                                    "updatedBy": "11",
                                    "archive": "false",
                                    "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
                                    "contentSections": [],
                                    "dynamicContents": [],
                                    "emailFooterId": "1",
                                    "emailGroupId": "4",
                                    "emailHeaderId": "1",
                                    "encodingId": "3",
                                    "fieldMerges": [],
                                    "forms": [],
                                    "htmlContent": {
                                    "type": "RawHtmlContent",
                                    "contentSource": "upload",
                                    "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <bod
<p>Test Low Volume Email</p>\r\n  </body> \r\n</html>"
                                    },
                                    "hyperlinks": [],
                                    "images": [],
                                    "isContentProtected": "false",
                                    "isPlainTextEditable": "false",
                                    "isPrivate": "False",
                                    "isTracked": "true",
                                    "layout": "{}",
                                    "plainText": "\r\nTest Low Volume Email\r\n\r\n",
                                    "renderMode": "Flow",
                                    "replyToEmail": "api.user@test.oracle.com",
                                    "replyToName": "API User",
                                    "sendPlainTextOnly": "false",
```

```
                    "senderEmail": "api.user@test.oracle.com",
                    "senderName": "API User",
                    "style": "{}",
                    "subject": "Test Low Volume"
                    },
                    "failedSendCount": "0",
                    "successfulSendCount": "0",
                    "contactIds": [
                    "1",
                    "2"
                    ],
                    "notificationEmailAddress": "api.user@test.oracle.com",
                    "sendDate": "1468622700",
                    "sendFromUserId": "11",
                    "sendOptions": {
                    "type": "EmailSendOptions",
                    "allowResend": "true",
                    "allowSendToBounceback": "false",
                    "allowSendToGroupUnsubscribe": "true",
                    "allowSendToMasterExclude": "false",
                    "allowSendToUnsubscribe": "false"
                    },
                    "signatureRuleId": "1"
                }
```


GET `api/REST/2.0/assets/email/deployment/{id}`

Retrieve an email deployment sent to a small number of contacts.

Request parameters

Required

Name	Type	Description	Possible values
id	integer	Unique identifier of the email deployment.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will be complete. Learn more about the depth parameter.	• `minimal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment	`EmailLowVolumeDeployment` **Note**: The activity's email send type will appear as `LowVolumeAPI`.
name	string	The name of the deployment.	
contactIds	array	The array of contact ids of the contacts to the deployment will attempt to send to. Contacts that were filtered out during deployment creation because of `sendOptions` are	

Name	Type	Description	Possible values
		removed from this array and are included in the exclusions array.	
exclusions	array	The array of errors describing the reason each contact was excluded from the deployment.	
successfulSendCount	string	The number of emails which have been successfully sent so far.	
failedSendCount	string	The number of emails which failed during send so far.	
currentStatus	string	Current deployment status.	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
sendFromUserId	string	The ID of the user used as the sender of the email. **Note**: If `signatureRuleId` is set, `sendFromUserId` will display the user id of the user	

Name	Type	Description	Possible values
		who sent the request.	
signatureRuleId	string	The ID of the signature rule used if specified on creation.	
permissions	array	Permission values of the deployment granted to the current user.	
depth	string	The request's level of detail.	• `minmal` • `partial` • `complete`

Nested Schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset	

Note: For a full list of parameters returned within the email object see Retrieve an email response parameters.

Nested Schema: sendOptions

Name	Type	Description	Possible values
type	string	The type of send options.	`EmailSendOptions`
allowResend	string	Whether or not this email is allowed to be resent to the contact.	
allowSendToMasterExclude	string	Whether or not this email is allowed to be sent to contacts in the master exclude list.	
allowSendToUnsubscribe	string	Whether or not this email is allowed to be sent to globally unsubscribed contacts	
allowSendToGroupUnsubscribe	string	Whether or not this email is allowed to be sent to contacts unsubscribed from the	

Name	Type	Description	Possible values
		email group	
allowSendToBounceback	string	Whether or not this email is allowed to be sent to contacts that are marked as having bounced.	

Example

Retrieve an email deployment with the id = 5 (before the scheduled send time):

```
GET /assets/email/deployment/5
```

Response:

```
{
  "type": "EmailLowVolumeDeployment",
  "currentStatus": "normal",
  "id": "5",
  "depth": "complete",
  "name": "REST Low Volume 01",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update",
    "Activate"
  ],
  "email": {
    "type": "Email",
    "currentStatus": "Draft",
    "id": "101",
    "createdAt": "1468621285",
```

```
  "createdBy": "11",
  "depth": "complete",
  "folderId": "42",
  "name": "Test_Low_Volume_Email",
  "permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update"
  ],
  "updatedAt": "1468621363",
  "updatedBy": "11",
  "archive": "false",
  "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
  "contentSections": [],
  "dynamicContents": [],
  "emailFooterId": "1",
  "emailGroupId": "4",
  "emailHeaderId": "1",
  "encodingId": "3",
  "fieldMerges": [],
  "forms": [],
  "htmlContent": {
   "type": "RawHtmlContent",
   "contentSource": "upload",
   "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <body>\r\n
<p>Test Low Volume Email</p>\r\n  </body> \r\n</html>"
  },
  "hyperlinks": [],
  "images": [],
  "isContentProtected": "false",
  "isPlainTextEditable": "false",
  "isPrivate": "False",
  "isTracked": "true",
  "layout": "{}",
  "plainText": "\r\nTest Low Volume Email\r\n\r\n",
  "renderMode": "Flow",
  "replyToEmail": "api.user@test.oracle.com",
  "replyToName": "API User",
  "sendPlainTextOnly": "false",
```

```
    "senderEmail": "api.user@test.oracle.com",
    "senderName": "API User",
    "style": "{}",
    "subject": "Test Low Volume"
  },
  "failedSendCount": "0",
  "successfulSendCount": "0",
  "contactIds": [
    "1",
    "2"
  ],
  "exclusions": [],
  "notificationEmailAddress": "api.user@test.oracle.com",
  "sendDate": "1468622700",
  "sendFromUserId": "11",
  "sendOptions": {
    "type": "EmailSendOptions",
    "allowResend": "true",
    "allowSendToBounceback": "false",
    "allowSendToGroupUnsubscribe": "true",
    "allowSendToMasterExclude": "false",
    "allowSendToUnsubscribe": "false"
  },
  "signatureRuleId": "1"
}
```


Retrieve an email deployment with the id = 5: (After send time)

```
GET /assets/email/deployment/5
```

Response:


```
{
  "type": "EmailLowVolumeDeployment",
  "currentStatus": "normal",
  "id": "5",
  "depth": "complete",
  "name": "REST Low Volume 01",
```

```
  "permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update",
   "Activate"
  ],
  "email": {
   "type": "Email",
   "currentStatus": "Draft",
   "id": "101",
   "createdAt": "1468621285",
   "createdBy": "11",
   "depth": "complete",
   "folderId": "42",
   "name": "Test_Low_Volume_Email",
   "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update"
   ],
   "updatedAt": "1468621363",
   "updatedBy": "11",
   "archive": "false",
   "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
   "contentSections": [],
   "dynamicContents": [],
   "emailFooterId": "1",
   "emailGroupId": "4",
   "emailHeaderId": "1",
   "encodingId": "3",
   "fieldMerges": [],
   "forms": [],
   "htmlContent": {
    "type": "RawHtmlContent",
    "contentSource": "upload",
    "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <body>\r\n
<p>Test Low Volume Email</p>\r\n  </body> \r\n</html>"
   },
```

```
    "hyperlinks": [],
    "images": [],
    "isContentProtected": "false",
    "isPlainTextEditable": "false",
    "isPrivate": "False",
    "isTracked": "true",
    "layout": "{}",
    "plainText": "\r\nTest Low Volume Email\r\n\r\n",
    "renderMode": "Flow",
    "replyToEmail": "api.user@test.oracle.com",
    "replyToName": "API User",
    "sendPlainTextOnly": "false",
    "senderEmail": "api.user@test.oracle.com",
    "senderName": "API User",
    "style": "{}",
    "subject": "Test Low Volume"
  },
  "endAt": "1468622701",
  "failedSendCount": "0",
  "successfulSendCount": "2",
  "contactIds": [
    "1",
    "2"
  ],
  "exclusions": [],
  "notificationEmailAddress": "api.user@test.oracle.com",
  "sendDate": "1468622700",
  "sendFromUserId": "11",
  "sendOptions": {
    "type": "EmailSendOptions",
    "allowResend": "true",
    "allowSendToBounceback": "false",
    "allowSendToGroupUnsubscribe": "true",
    "allowSendToMasterExclude": "false",
    "allowSendToUnsubscribe": "false"
  },
  "signatureRuleId": "1"
}
```


Retrieve an email deployment with the id = 5 at minimal depth:

```
GET /assets/email/deployment/5?depth=minimal
```

Response:


```
{
  "type": "EmailLowVolumeDeployment",
  "id": "5",
  "depth": "minimal",
  "name": "REST Low Volume 01",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update",
    "Activate"
  ],
  "failedSendCount": "0",
  "successfulSendCount": "2",
  "exclusions": []
}
```


Retrieve an email deployment with the ID = 5 at partial depth:

```
GET /assets/email/deployment/5?depth=partial
```

Response:


```
{
  "type": "EmailLowVolumeDeployment",
  "currentStatus": "normal",
  "id": "5",
  "depth": "partial",
  "name": "REST Low Volume 01",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
```

```
    "Update",
    "Activate"
  ],
  "email": {
    "type": "Email",
    "currentStatus": "Draft",
    "id": "101",
    "createdAt": "1468621285",
    "createdBy": "11",
    "depth": "partial",
    "folderId": "42",
    "name": "Test_Low_Volume_Email",
    "permissions": [
      "Retrieve",
      "SetSecurity",
      "Delete",
      "Update"
    ],
    "updatedAt": "1468621363",
    "updatedBy": "11",
    "archive": "false",
    "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
    "emailFooterId": "1",
    "emailGroupId": "4",
    "emailHeaderId": "1",
    "encodingId": "3",
    "fieldMerges": [],
    "htmlContent": {
      "type": "RawHtmlContent",
      "contentSource": "upload"
    },
    "hyperlinks": [],
    "isContentProtected": "false",
    "isPlainTextEditable": "false",
    "isPrivate": "False",
    "isTracked": "true",
    "layout": "{}",
    "plainText": "\r\nTest Low Volume Email\r\n\r\n",
    "renderMode": "Flow",
    "replyToEmail": "api.user@test.oracle.com",
```

```
    "replyToName": "API User",
    "sendPlainTextOnly": "false",
    "senderEmail": "api.user@test.oracle.com",
    "senderName": "API User",
    "style": "{}",
    "subject": "Test Low Volume"
  },
  "endAt": "1468622701",
  "failedSendCount": "0",
  "successfulSendCount": "2",
  "contactIds": [
    "1",
    "2"
  ],
  "exclusions": [],
  "notificationEmailAddress": "api.user@test.oracle.com",
  "sendDate": "1468622700",
  "sendFromUserId": "11",
  "sendOptions": {
    "type": "EmailSendOptions",
    "allowResend": "true",
    "allowSendToBounceback": "false",
    "allowSendToGroupUnsubscribe": "true",
    "allowSendToMasterExclude": "false",
    "allowSendToUnsubscribe": "false"
  },
  "signatureRuleId": "1"
}
```


POST `api/REST/2.0/assets/email/deployment`

This endpoint is used to create and send an HTML email deployment to up to 2000 contacts.

Request parameters

Required

Name	Type	Description	Possible values
type	string	The type of email deployment.	• `EmailInlineDeployment` Sends an HTML email to up to 2000 contacts **Note**: The activity's email send type will appear as `SalesTools`.
name	string	The name of the deployment.	
contacts	array	The array of contact IDs of the contacts to send to.	

Nested schema: `email`

Details about the HTML email deployment.

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`Email`
name	string	The name of the email asset. **Note**: The name is not actually read so any value can be used but it is still required.	

Name	Type	Description	Possible values
subject	string	The email subject.	
id	string	(Optional) The email ID to specify if you want to send an existing email deployment.	

Nested schema: `htmlContent`

The HTML content for the email deployment.

Name	Type	Description	Possible values
type	string	The email type. Must be `RawHtmlContent`.	`RawHtmlContent`
html	string	The raw HTML content for the email.	

Optional

Name	Type	Description	Possible values
sendFromUserId	string	The ID of the user to use as the sender of the email. The default will use the current user.	
sendDate	string	A date in the future to schedule the deployment. If the date is in the past, or not specified, the deployment is scheduled immediately. The format is Unix Time.	
externalSource	string	Indicates the source of the email. If not specified, set to `Unknown`.	• `Unknown` • `Engage` • `SalesCloud`
depth	string	Level of detail returned by the request. The default will be complete. Learn more about the depth parameter.	• `minimal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment.	`EmailInlineDeployment` **Note**: The activity's Email Send Type will appear as `EmailInlineDeployment`.
currentStatus	string	Current deployment status.	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
id	string	The ID of the email asset	
depth	string	The request's level of detail.	`complete`
name	string	The name of the deployment.	
permissions	array	Permission values of the deployment granted to the current user.	

Name	Type	Description	Possible values
successfulSendCount	string	The number of emails which have been successfully sent so far.	
failedSendCount	string	The number of emails which failed during send so far.	
sendDate	string	The date the deployment was scheduled for send. The format is Unix Time. If scheduled for a future date, the email will be deployed at the future date and time. Otherwise the email will	

Name	Type	Description	Possible values
		be sent right away.	
sentContent	string	The contents of the email sent.	
sentSubject	string	The email subject.	
successfulSendCount	string	The amount of email deployments sent successfully to contacts.	
clickthroughCount	string	The count of user clickthroughs recorded on the email.	
contacts	array	The array of contact ids of the contacts to the deployment will attempt to send to.	
externalSource	string	Indicates the source of the email. If not specified, set to `Unknown`.	• `Unknown` • `Engage` • `SalesCloud`
openCount	string	The count of user email opens for the email.	
sendFromUserId	string	The ID of the	

Name	Type	Description	Possible values
		user used as the sender of the email.	
statistics	array	Statistics about the email deployment such as bounceback type, clickthrough count, email open date, and email send date. Retrieve an email deployment to see email deployment statistics.	

Nested schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`Email`
id	string	The ID of the email asset.	
name	string	The name	

Name	Type	Description	Possible values
		of the email asset.	

Nested schema: `htmlContent`

The HTML content for the email deployment.

Name	Type	Description	Possible values
type	string	The email type. Must be `RawHtmlContent`.	`RawHtmlContent`
html	string	The raw HTML content for the email.	

Example

Create and send an email deployment for an HTML email to two contacts:

```
POST /assets/email/deployment
```

Request body:

```
{
  "type": "EmailInlineDeployment",
  "depth": "complete",
  "name": "Email Send Future Date",
  "email": {
    "type": "Email",
    "name": "Test Inline Deployment",
    "subject": "Email Deployment Test",
    "htmlContent": {
      "type": "RawHtmlContent",
      "html": "<html><head></head><body>Test Inline Deployment</body></html>"
    }
  },
```

```
  "sendDate": "1601363751",
  "externalSource": "SalesCloud",
  "sendFromUserId": 2,
  "contacts": [
   {
    "id": 3,
    "id": 4
   }
  ]
}
```

Response:

```
{
  "type": "EmailInlineDeployment",
  "currentStatus": "normal",
  "id": "209",
  "depth": "complete",
  "name": "Email Send Future Date",
  "permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update",
   "Activate"
  ],
  "email": {
   "type": "Email",
   "replyToEmail": "newclient@oracle.com",
   "replyToName": "New Client",
   "senderEmail": "newclient@oracle.com",
   "senderName": "New Client"
  },
  "failedSendCount": "0",
  "sendDate": "1601363751",
  "sentContent": "<html><head></head><body>Test Inline Deployment</body></html>",
  "sentSubject": "Email Deployment Test",
  "successfulSendCount": "0",
  "clickthroughCount": "0",
```

```
  "contacts": [],
  "externalSource": "SalesCloud",
  "openCount": "0",
  "sendFromUserId": "2",
  "statistics": []
}
```

GET `api/REST/2.0/assets/email/deployment/{id}`

Retrieve an email deployment sent to a large number of contacts.

Request parameters

Required

Name	Type	Description	Possible values
id	integer	Unique identifier of the email deployment.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will be complete. Learn more about the depth parameter.	• `minimal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of email deployment.	`EmailInlineDeployment` **Note**: The activity's Email Send Type will appear as `SalesTools`.
currentStatus	string	Current deployment status.	• `normal` • `locked` • `forceComplete` • `inError` • `resend`
id	string	The ID of the email asset	
depth	string	The request's level of detail.	`complete`
name	string	The name of the deployment.	
permissions	array	Permission values of the deployment granted to the current user.	
successfulSendCount	string	The number of emails which have been successfully sent	

Name	Type	Description	Possible values
		so far.	
failedSendCount	string	The number of emails which failed during send so far.	
sendDate	string	The date the deployment was scheduled for send. The format is Unix Time. If scheduled for a future date, the email will be deployed at the future date and time. Otherwise the email will be sent right away.	
sentContent	string	The contents of the email sent.	
sentSubject	string	The email subject.	
successfulSendCount	string	The amount of email deployments sent successfully to contacts.	
clickthroughCount	string	The count of user	

Name	Type	Description	Possible values
		clickthroughs recorded on the email.	
contacts	array	The array of contact ids of the contacts to the deployment will attempt to send to. **Note**: For a full list of parameters returned within the contact object see Retrieve a contact response parameters.	
externalSource	string	Indicates the source of the	• `Unknown`

Name	Type	Description	Possible values
		email. If not specified, set to `Unknown`.	• `Engage` • `SalesCloud`
openCount	string	The count of user email opens for the email.	
sendFromUserId	string	The ID of the user used as the sender of the email.	
statistics	array	Statistics about the email deployment such as bounceback type, clickthrough count, email open date, and email send date.	

Nested Schema: email

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua	`Email`
id	string	The ID of the email asset	
name	string	The name of the email asset	

Note: For a full list of parameters returned within the email object see Retrieve an email response parameters.

Example

Retrieve an email deployment with the id = 10:

```
GET /assets/email/deployment/10
```

Response:


```
{
  "type": "EmailInlineDeployment",
  "currentStatus": "normal",
  "id": "10",
  "depth": "complete",
  "name": "Email Send Future Date",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update",
    "Activate"
  ],
  "email": {
    "type": "Email",
    "replyToEmail": "api.user@oracle.com",
    "replyToName": "New Client",
    "senderEmail": "api.user@oracle.com",
    "senderName": "New Client"
  },
  "endAt": "1604439829",
  "failedSendCount": "0",
  "sentContent": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n  <body>\r\n
<p>Test Low Volume Email</p>\r\n  </body> \r\n</html>",
```

```
  "sentSubject": "test email send",
  "successfulSendCount": "1",
  "clickthroughCount": "0",
  "contacts": [
   {
    "type": "Contact",
    "currentStatus": "Awaiting action",
    "id": "4",
    "createdAt": "1424720465",
    "depth": "partial",
    "name": "api.user@oracle.com",
    "updatedAt": "1600783263",
    "emailAddress": "api.user@oracle.com",
    "emailFormatPreference": "unspecified",
    "firstName": "api",
    "isBounceback": "false",
    "isSubscribed": "true",
    "lastName": "user",
    "subscriptionDate": "1424720465"
   }
  ],
  "openCount": "1",
  "sendFromUserId": "2",
  "statistics": [
   {
    "type": "EmailDeploymentStatistic",
    "bouncebackType": "",
    "clickthroughCount": "3",
    "contactId": "4",
    "emailAddress": "api.user@oracle.com",
    "openCount": "1",
    "sentAt": "1604439828"
   }
  ]
}
```

Signature rules API

Overview

With signature rules, emails can be sent on behalf of many different people at once in a batch email deployment. The senders are dynamically defined based on the recipients of the message. Rules link a specific sender to a contact based on the field values in that contact's record or custom object.

Learn more about signature rules.

API Tasks

Retrieve a signature rule

Retrieve a list of signature rules

GET `api/REST/2.0/assets/email/signature/rule/{id}`

Retrieves the signature rule asset specified by the `{id}` parameter.

Request parameters

Required

Name	Type	Description	Possible values
id	integer	Unique identifier of the email signature rule.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will	• `minmal`

Name	Type	Description	Possible values
		be complete. Learn more about the depth parameter.	• `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`EmailSignatureRule`
id	string	Id of the signature rule.	
createdAt	string	The date and time the email signature rule was created.	
createdBy	string	The login id of the user who created the signature rule.	
depth	string	The request's level of detail.	• `minmal` • `partial` • `complete`

Name	Type	Description	Possible values
name	string	The name of the signature rule.	
updatedAt	string	Unix timestamp for the date and time the signature rule was last updated.	
updatedBy	string	The login id of the user that last updated the signature rule.	
contactFieldId	string	The ID of the field used to identify the email sender	
defaultSenderId	string	The ID of the user used as the sender of the email	
isPersonalizeFromAddress	string	Whether or not the email sender's "From"	

Name	Type	Description	Possible values
		address is customized in the email header	
isPersonalizeFromName	string	Whether or not the email sender's "Display Name" is customized in the email header	
isPersonalizeReplyAddress	string	Whether or not the "Reply-To Address" is customized in the email header	
isPersonalizeReplyName	string	Whether or not the "Reply-To Display Name" is customized in the email header	

Example

Retrieve a signature rule with the id = 5.

```
GET /assets/email/signature/rule/5
```

Response:


```
{
  "type": "EmailSignatureRule",
  "id": "2",
  "createdAt": "1417724000",
  "createdBy": "12",
  "depth": "complete",
  "name": "EC Test Signature Rule",
  "updatedAt": "1471936138",
  "updatedBy": "71",
  "contactFieldId": "100022",
  "defaultSenderId": "6",
  "isPersonalizeFromAddress": "True",
  "isPersonalizeFromName": "True",
  "isPersonalizeReplyAddress": "True",
  "isPersonalizeReplyName": "True"
}
```


GET `api/REST/2.0/assets/email/signature/rules`

Retrieves a list of signature rule assets.

Request parameters

Name	Description	Constraints
`depth`	Level of detail returned by the request. Learn more about the depth parameter.	**Possible values:** "minimal" "partial" "complete" **Example:** `?depth=complete`
`count`	Maximum number of entities to return	Any whole number between 1 and 1000 inclusive.

Name	Description	Constraints
		Example: `?count=100`
`page`	Specifies which page of entities to return (the count parameter defines the number of entities per page). If the page parameter is not supplied, 1 will be used by default.	Any positive whole number. **Example:** `?page=3&count=10`
`search`	The `search` parameter specifies the search criteria to use to filter the results. The syntax for the search parameter is: `search={term}{operator}{value}`. Learn more about the search URL parameter.	`{term}` is the name of a field or property to filter on, `{operator}` is the comparison operator, and `{value}` is the value to compare the term with. If `{term}` and `{operator}` are not supplied, the term is compared to the value using the equality operator. Searches can be for exact full matches, or partial matches. A "*****" symbol can be used after the `{value}` for partial matches. If there are spaces in the `value`, the `value` needs to be placed in single quotes. Otherwise, single quotes are not required.

Name	Description	Constraints
		You can search with fields even if they are not returned at the depth being used. The following operators are supported on most endpoints: • **=** (Equal To) • **!=** (Not equal to) • **>** (Greater than) • **<** (Less than) • **>=** (Greater than or Equal to) • **<=** (Less than or Equal to) **Example**: `GET .../data/contacts?search=id=1`
`sort`	Specifies the name of the property used to sort the returned entities.	The value depends on the type of entity being sorted, with each entity having its own list of sortable properties. **Example**: `GET .../data/contacts?sort=firstName`
`dir`	Specifies the direction in which to sort the returned entities.	"asc" for ascending or "desc" for descending. **Example**: `GET`

Name	Description	Constraints
		`.../data/contacts?sort=firstName&dir=asc`

`orderBy`	Specifies the field by which list results are ordered, and the direction. The direction will defaullt to ASC if not specified.	Any valid asset parameter field.
		Example:
		`?orderBy=createdAt`
		`?orderBy=createdAt DESC`
		`?orderBy=createdAt ASC`

`lastUpdatedAt`	When the asset was last updated. Returns deleted assets.**Note**: For the majority of use cases, it is recommended to use `updatedAt` with the `search` URL parameter. For example: `?search='updatedAt>518862600'`	A valid date/time value. **Example**: `?lastUpdatedAt=518862600`

Response parameters

Name	Type	Description	Possible values
type	string	The asset's type in Eloqua.	`EmailSignatureRule`
id	string	Id of the signature	

Name	Type	Description	Possible values
		rule.	
createdAt	string	The date and time the signature rule was created.	
createdBy	string	The login id of the user who created the signature rule.	
depth	string	The request's level of detail.	• `minmal` • `partial` • `complete`
name	string	The name of the signature rule.	
updatedAt	string	Unix timestamp for the date and time the signature rule was last updated.	
updatedBy	string	The login id of the user that last updated the	

Name	Type	Description	Possible values
		signature rule.	
isPersonalizeFromAddress	string	Whether or not the email sender's "From" address is customized in the email header.	
isPersonalizeFromName	string	Whether or not the email sender's "Display Name" is customized in the email header.	
isPersonalizeReplyAddress	string	Whether or not the "Reply-To Address" is customized in the email header.	
isPersonalizeReplyName	string	Whether or not the "Reply-To Display Name" is customized in the email header.	

Example

Retrieve the first two signature rules in your database:

```
GET /assets/email/signature/rules?count=2
```

Response:


```
{
  "elements": [
    {
      "type": "EmailSignatureRule",
      "id": "2",
      "createdAt": "1417724000",
      "createdBy": "12",
      "depth": "minimal",
      "name": "EC Test Signature Rule",
      "updatedAt": "1471936138",
      "updatedBy": "71",
      "isPersonalizeFromAddress": null,
      "isPersonalizeFromName": null,
      "isPersonalizeReplyAddress": null,
      "isPersonalizeReplyName": null
    },
    {
      "type": "EmailSignatureRule",
      "id": "115",
      "createdAt": "1460656435",
      "createdBy": "2",
      "depth": "minimal",
      "name": "rl 480 Signature Rule",
      "updatedAt": "1460656861",
      "updatedBy": "2",
      "isPersonalizeFromAddress": null,
      "isPersonalizeFromName": null,
      "isPersonalizeReplyAddress": null,
      "isPersonalizeReplyName": null
    }
  ],
```

```
  "page": 1,
  "pageSize": 2,
  "total": 96
}
```

POST `api/REST/2.0/data/accounts`

Retrieves account information for up to 200 accounts, searched by account id. Use the `depth` parameter when making the request to specify the account information returned in the response. Note that for accounts, `minimal` and `partial` depths return the same information.

> **Important**: Request header 'X-HTTP-Method-Override' needs to be set with value 'SEARCH'.

Request parameters

Required

Name	Type	Description	Possible values
ids	array	The account ids to retrieve. Maximum of 200 account ids per request. If the array contains duplicate account ids, or exceeds the 200 limit, a 400 Bad Request will be returned.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will be minimal. Learn more about the depth parameter. **Note**: For accounts, `minimal` and `partial` depths return the same information.	• `minimal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of asset	`Account`

Name	Type	Description	Possible values
		in Eloqua.	
id	string	The account id of the account.	
createdAt	string	The date and time the account was created, expressed in Unix time.	
depth	string	The request's level of detail. **Note**: For accounts, `minimal` and `partial` depths return the same information.	• `minimal` • `partial` • `complete`
description	string	The description of the account.	
name	string	The name of the account.	
updatedAt	string	The date and time the contact was	

Name	Type	Description	Possible values
		last updated, expressed in Unix time.	
address1	string	The account's first address.	
city	string	The account's city.	

Note: For a full list of parameters returned within the account object see Retrieve an account response parameters.

Example

Retrieve 3 accounts at minimal depth:

```
POST api/REST/2.0/data/accounts
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
   "ids": [2, 3, 5],
   "depth": "minimal"
}
```

Response:

```
[
 {
  "type": "Account",
  "id": "2",
  "createdAt": "1462393916",
  "depth": "minimal",
  "description": "",
  "name": "Initech",
  "updatedAt": "1462393916",
  "address1": "101 Mainstreet",
  "city": "Austin",
  "fieldValues": [
   {
    "type": "FieldValue",
    "id": "100102"
   },
   {
    "type": "FieldValue",
    "id": "100094",
    "value": "MDC13000000000002"
   }
  ],
  "province": "TX"
 },
 {
  "type": "Account",
  "id": "3",
  "createdAt": "1462395731",
  "depth": "minimal",
  "description": "",
  "name": "An Example Company",
  "updatedAt": "1462395731",
  "address1": "Business Blvd",
  "businessPhone": "1111111111",
  "city": "Deluge",
  "country": "US",
  "fieldValues": [
   {
    "type": "FieldValue",
    "id": "100102"
```

```
      },
      {
        "type": "FieldValue",
        "id": "100094",
        "value": "MDC13000000000003"
      }
    ],
    "postalCode": "MR4 5UX",
    "province": "CT"
  },
  {
    "type": "Account",
    "id": "5",
    "createdAt": "1462482690",
    "depth": "minimal",
    "description": "",
    "name": "Globex",
    "updatedAt": "1462482690",
    "address1": "555 Downtown Lane",
    "city": "Denver",
    "fieldValues": [
      {
        "type": "FieldValue",
        "id": "100102"
      },
      {
        "type": "FieldValue",
        "id": "100094",
        "value": "MDC13000000000005"
      }
    ],
    "province": "Colorado"
  }
]
```


Retrieve 3 accounts at partial depth:

```
POST api/REST/2.0/data/accounts
X-HTTP-Method-Override: SEARCH
```

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

Request body:

```
{
   "ids": [2, 3, 5],
   "depth": "partial"
}
```

Response:

```
[
 {
   "type": "Account",
   "id": "2",
   "createdAt": "1462393916",
   "depth": "partial",
   "description": "",
   "name": "Initech",
   "updatedAt": "1462393916",
   "address1": "101 Mainstreet",
   "city": "Austin",
   "fieldValues": [
    {
      "type": "FieldValue",
      "id": "100102"
    },
    {
      "type": "FieldValue",
      "id": "100094",
      "value": "MDC130000000000002"
    }
   ],
   "province": "TX"
 },
 {
   "type": "Account",
   "id": "3",
```

```
  "createdAt": "1462395731",
  "depth": "partial",
  "description": "",
  "name": "An Example Company",
  "updatedAt": "1462395731",
  "address1": "Business Blvd",
  "businessPhone": "1111111111",
  "city": "Deluge",
  "country": "US",
  "fieldValues": [
    {
      "type": "FieldValue",
      "id": "100102"
    },
    {
      "type": "FieldValue",
      "id": "100094",
      "value": "MDC13000000000003"
    }
  ],
  "postalCode": "MR4 5UX",
  "province": "CT"
},
{
  "type": "Account",
  "id": "5",
  "createdAt": "1462482690",
  "depth": "partial",
  "description": "",
  "name": "Globex",
  "updatedAt": "1462482690",
  "address1": "555 Downtown Lane",
  "city": "Denver",
  "fieldValues": [
    {
      "type": "FieldValue",
      "id": "100102"
    },
    {
      "type": "FieldValue",
```

```
      "id": "100094",
      "value": "MDC13000000000005"
    }
   ],
   "province": "Colorado"
  }
]
```

Retrieve 3 accounts at complete depth:

```
POST api/REST/2.0/data/accounts
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
   "ids": [2, 3, 5],
   "depth": "complete"
}
```

Response:

```
[
 {
   "type": "Account",
   "id": "2",
   "createdAt": "1462393916",
   "createdBy": "1462393916",
   "depth": "complete",
   "description": "",
   "name": "Initech",
   "updatedAt": "1462393916",
   "address1": "101 Mainstreet",
   "city": "Austin",
   "fieldValues": [
    {
```

```
      "type": "FieldValue",
      "id": "100102"
    },
    {
      "type": "FieldValue",
      "id": "100094",
      "value": "MDC13000000000002"
    },
    {
      "type": "FieldValue",
      "id": "100097"
    },
    {
      "type": "FieldValue",
      "id": "100100"
    },
    {
      "type": "FieldValue",
      "id": "100119"
    },
    {
      "type": "FieldValue",
      "id": "100170"
    },
    {
      "type": "FieldValue",
      "id": "100189"
    }
  ],
  "province": "TX"
},
{
  "type": "Account",
  "id": "3",
  "createdAt": "1462395731",
  "createdBy": "1462395731",
  "depth": "complete",
  "description": "",
  "name": "An Example Company",
  "updatedAt": "1462395731",
```

```
  "address1": "Business Blvd",
  "businessPhone": "1111111111",
  "city": "Deluge",
  "country": "US",
  "fieldValues": [
   {
    "type": "FieldValue",
    "id": "100102"
   },
   {
    "type": "FieldValue",
    "id": "100094",
    "value": "MDC13000000000003"
   },
   {
    "type": "FieldValue",
    "id": "100097"
   },
   {
    "type": "FieldValue",
    "id": "100100"
   },
   {
    "type": "FieldValue",
    "id": "100119"
   },
   {
    "type": "FieldValue",
    "id": "100170",
    "value": "6666666666"
   },
   {
    "type": "FieldValue",
    "id": "100189"
   }
  ],
  "postalCode": "MR4 5UX",
  "province": "CT"
},
{
```

```
"type": "Account",
"id": "5",
"createdAt": "1462482690",
"createdBy": "1462482690",
"depth": "complete",
"description": "",
"name": "Globex",
"updatedAt": "1462482690",
"address1": "555 Downtown Lane",
"city": "Denver",
"fieldValues": [
 {
  "type": "FieldValue",
  "id": "100102"
 },
 {
  "type": "FieldValue",
  "id": "100094",
  "value": "MDC13000000000005"
 },
 {
  "type": "FieldValue",
  "id": "100097"
 },
 {
  "type": "FieldValue",
  "id": "100100"
 },
 {
  "type": "FieldValue",
  "id": "100119"
 },
 {
  "type": "FieldValue",
  "id": "100170"
 },
 {
  "type": "FieldValue",
  "id": "100189"
 }
```

```
    ],
    "province": "Colorado"
  }
]
```

POST `api/REST/2.0/data/contacts`

Retrieves contact information for up to 200 contacts, searched by contact id. Use the `depth` parameter when making the request to specify the contact information returned in the response.

> **Important**: Request header 'X-HTTP-Method-Override' needs to be set with value 'SEARCH'.

Request parameters

Required

Name	Type	Description	Possible values
ids	array	The contact ids to retrieve. Maximum of 200 contact ids per request. If the array contains duplicate contact ids, or exceeds the 200 limit, a 400 Bad Request will be returned.	

Optional

Name	Type	Description	Possible values
depth	string	Level of detail returned by the request. The default will be minimal. Learn more about the depth parameter.	• `minmal` • `partial` • `complete`

Response parameters

Name	Type	Description	Possible values
type	string	The type of asset in Eloqua.	`Contact`
id	string	The contact id of the contact.	
createdAt	string	The date and time the contact was created,	

Name	Type	Description	Possible values
		expressed in Unix time.	
depth	string	The request's level of detail.	• `minmal` • `partial` • `complete`
name	string	The name of the contact.	
updatedAt	string	The date and time the contact was last updated, expressed in Unix time.	

Note: For a full list of parameters returned within the contact object see Retrieve a contact response parameters.

Example

Retrieve 3 contacts at minimal depth:

```
POST api/REST/2.0/data/contacts
X-HTTP-Method-Override: SEARCH
```

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

Request body:

```
{
   "ids": [1, 2, 3],
   "depth": "minimal"
}
```

Response:

```
[{
   "type": "Contact",
   "id": "1",
   "createdAt": "1418667629",
   "depth": "minimal",
   "name": "api.user@test.oracle.com",
   "updatedAt": "1466689992"
}, {
   "type": "Contact",
   "id": "2",
   "createdAt": "1418673492",
   "depth": "minimal",
   "name": "api.user2@test.oracle.com",
   "updatedAt": "1469552212"
}, {
   "type": "Contact",
   "id": "3",
   "createdAt": "1420586365",
   "depth": "minimal",
   "name": "api.user3@test.oracle.com",
   "updatedAt": "1466689992"
}]
```

Retrieve 3 contacts at partial depth:

```
POST api/REST/2.0/data/contacts
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
   "ids": [1, 2, 3],
   "depth": "partial"
}
```

Response:

```
[
 {
   "type": "Contact",
   "currentStatus": "Awaiting action",
   "id": "1",
   "createdAt": "1418667629",
   "depth": "partial",
   "name": "api.user@test.oracle.com",
   "updatedAt": "1466689992",
   "emailAddress": "api.user@test.oracle.com",
   "emailFormatPreference": "unspecified",
   "firstName": "Test",
   "isBounceback": "false",
   "isSubscribed": "true",
   "lastName": "User",
   "subscriptionDate": "1418673391"
 },
 {
   "type": "Contact",
   "currentStatus": "Awaiting action",
   "id": "2",
   "createdAt": "1418673492",
   "depth": "partial",
   "name": "api.user2@test.oracle.com",
   "updatedAt": "1469552212",
```

```
    "bouncebackDate": "1469552198",
    "emailAddress": "api.user2@test.oracle.com",
    "emailFormatPreference": "unspecified",
    "firstName": "TEST",
    "isBounceback": "true",
    "isSubscribed": "true",
    "lastName": "USER",
    "subscriptionDate": "1418673492"
  },
  {
    "type": "Contact",
    "currentStatus": "Awaiting action",
    "id": "3",
    "createdAt": "1420586365",
    "depth": "partial",
    "name": "api.user3@test.oracle.com",
    "updatedAt": "1466689992",
    "emailAddress": "api.user3@test.oracle.com",
    "emailFormatPreference": "unspecified",
    "firstName": "Test",
    "isBounceback": "false",
    "isSubscribed": "true",
    "lastName": "User",
    "subscriptionDate": "1420586365"
  }
]
```


Retrieve 3 contacts at complete depth:

```
POST api/REST/2.0/data/contacts
X-HTTP-Method-Override: SEARCH
Content-Type: application/json
```

Request body:

```
{
    "ids": [1, 2, 3],
    "depth": "complete"
```

```
}
```

Response:


```
[
 {
  "type": "Contact",
  "currentStatus": "Awaiting action",
  "id": "1",
  "createdAt": "1418667629",
  "depth": "complete",
  "name": "api.user@test.oracle.com",
  "updatedAt": "1466689992",
  "emailAddress": "api.user@test.oracle.com",
  "emailFormatPreference": "unspecified",
  "fieldValues": [
   {
    "type": "FieldValue",
    "id": "100005"
   },
   {
    "type": "FieldValue",
    "id": "100017"
   },
   {
    "type": "FieldValue",
    "id": "100023"
   },
   {
    "type": "FieldValue",
    "id": "100024"
   },
   {
    "type": "FieldValue",
    "id": "100032",
    "value": "CDC13000000000001"
   },
   {
    "type": "FieldValue",
```

```
  "id": "100033"
 },
 {
  "type": "FieldValue",
  "id": "100034"
 },
 {
  "type": "FieldValue",
  "id": "100035"
 },
 {
  "type": "FieldValue",
  "id": "100036"
 },
 {
  "type": "FieldValue",
  "id": "100041"
 },
 {
  "type": "FieldValue",
  "id": "100043"
 },
 {
  "type": "FieldValue",
  "id": "100044"
 },
 {
  "type": "FieldValue",
  "id": "100045"
 },
 {
  "type": "FieldValue",
  "id": "100046"
 },
 {
  "type": "FieldValue",
  "id": "100047"
 },
 {
  "type": "FieldValue",
```

```
  "id": "100048"
 },
 {
  "type": "FieldValue",
  "id": "100049"
 },
 {
  "type": "FieldValue",
  "id": "100051"
 },
 {
  "type": "FieldValue",
  "id": "100065"
 },
 {
  "type": "FieldValue",
  "id": "100066"
 },
 {
  "type": "FieldValue",
  "id": "100068"
 },
 {
  "type": "FieldValue",
  "id": "100069"
 },
 {
  "type": "FieldValue",
  "id": "100072"
 },
 {
  "type": "FieldValue",
  "id": "100081"
 },
 {
  "type": "FieldValue",
  "id": "100171",
  "value": "test.oracle.com"
 },
 {
```

```
  "type": "FieldValue",
  "id": "100172",
  "value": "Test User1"
 },
 {
  "type": "FieldValue",
  "id": "100174"
 },
 {
  "type": "FieldValue",
  "id": "100175"
 },
 {
  "type": "FieldValue",
  "id": "100176"
 },
 {
  "type": "FieldValue",
  "id": "100177"
 },
 {
  "type": "FieldValue",
  "id": "100178"
 },
 {
  "type": "FieldValue",
  "id": "100179"
 },
 {
  "type": "FieldValue",
  "id": "100180"
 },
 {
  "type": "FieldValue",
  "id": "100181",
  "value": "da68a5a72dc49fa2001646dad425ba33"
 },
 {
  "type": "FieldValue",
  "id": "100182",
```

```
      "value":
"84ee430a8f7b234085611386a79ee9ccc604103b85e3d2da94a109b3b34da98b"
    },
    {
      "type": "FieldValue",
      "id": "100183"
    },
    {
      "type": "FieldValue",
      "id": "100184"
    },
    {
      "type": "FieldValue",
      "id": "100185"
    },
    {
      "type": "FieldValue",
      "id": "100186"
    },
    {
      "type": "FieldValue",
      "id": "100187"
    },
    {
      "type": "FieldValue",
      "id": "100188",
      "value": "TestUser1L88888"
    }
  ],
  "firstName": "Test",
  "isBounceback": "false",
  "isSubscribed": "true",
  "lastName": "User",
  "subscriptionDate": "1418673391"
 },
 {
  "type": "Contact",
  "currentStatus": "Awaiting action",
  "id": "2",
  "createdAt": "1418673492",
```

```
"depth": "complete",
"name": "api.user2@test.oracle.com",
"updatedAt": "1469552212",
"bouncebackDate": "1469552198",
"emailAddress": "api.user2@test.oracle.com",
"emailFormatPreference": "unspecified",
"fieldValues": [
 {
  "type": "FieldValue",
  "id": "100005"
 },
 {
  "type": "FieldValue",
  "id": "100017"
 },
 {
  "type": "FieldValue",
  "id": "100023"
 },
 {
  "type": "FieldValue",
  "id": "100024"
 },
 {
  "type": "FieldValue",
  "id": "100032",
  "value": "CDC13000000000002"
 },
 {
  "type": "FieldValue",
  "id": "100033"
 },
 {
  "type": "FieldValue",
  "id": "100034"
 },
 {
  "type": "FieldValue",
  "id": "100035"
 },
```

```
{
  "type": "FieldValue",
  "id": "100036"
},
{
  "type": "FieldValue",
  "id": "100041"
},
{
  "type": "FieldValue",
  "id": "100043"
},
{
  "type": "FieldValue",
  "id": "100044"
},
{
  "type": "FieldValue",
  "id": "100045"
},
{
  "type": "FieldValue",
  "id": "100046"
},
{
  "type": "FieldValue",
  "id": "100047"
},
{
  "type": "FieldValue",
  "id": "100048"
},
{
  "type": "FieldValue",
  "id": "100049"
},
{
  "type": "FieldValue",
  "id": "100051"
},
```

```
{
  "type": "FieldValue",
  "id": "100065"
},
{
  "type": "FieldValue",
  "id": "100066"
},
{
  "type": "FieldValue",
  "id": "100068"
},
{
  "type": "FieldValue",
  "id": "100069"
},
{
  "type": "FieldValue",
  "id": "100072"
},
{
  "type": "FieldValue",
  "id": "100081"
},
{
  "type": "FieldValue",
  "id": "100171",
  "value": "oracle.com"
},
{
  "type": "FieldValue",
  "id": "100172",
  "value": "TEST USER C"
},
{
  "type": "FieldValue",
  "id": "100174"
},
{
  "type": "FieldValue",
```

```
  "id": "100175"
 },
 {
  "type": "FieldValue",
  "id": "100176"
 },
 {
  "type": "FieldValue",
  "id": "100177"
 },
 {
  "type": "FieldValue",
  "id": "100178"
 },
 {
  "type": "FieldValue",
  "id": "100179"
 },
 {
  "type": "FieldValue",
  "id": "100180"
 },
 {
  "type": "FieldValue",
  "id": "100181",
  "value": "1591dd1bbaccf044659695773522aff0"
 },
 {
  "type": "FieldValue",
  "id": "100182",
  "value": "181bd0a7d97ba1b9b7475089b97bb91ba9eb84f2b195ec32a50f8cf39fb973ad"
 },
 {
  "type": "FieldValue",
  "id": "100183"
 },
 {
  "type": "FieldValue",
  "id": "100184"
 },
```

```
    {
      "type": "FieldValue",
      "id": "100185"
    },
    {
      "type": "FieldValue",
      "id": "100186"
    },
    {
      "type": "FieldValue",
      "id": "100187"
    },
    {
      "type": "FieldValue",
      "id": "100188",
      "value": "TESTUSERCL8888L"
    }
  ],
  "firstName": "Test",
  "isBounceback": "true",
  "isSubscribed": "true",
  "lastName": "User",
  "subscriptionDate": "1418673492"
},
{
  "type": "Contact",
  "currentStatus": "Awaiting action",
  "id": "3",
  "createdAt": "1420586365",
  "depth": "complete",
  "name": "api.user3@test.oracle.com",
  "updatedAt": "1466689992",
  "emailAddress": "api.user3@test.oracle.com",
  "emailFormatPreference": "unspecified",
  "fieldValues": [
    {
      "type": "FieldValue",
      "id": "100005"
    },
    {
```

```
  "type": "FieldValue",
  "id": "100017"
 },
 {
  "type": "FieldValue",
  "id": "100023"
 },
 {
  "type": "FieldValue",
  "id": "100024"
 },
 {
  "type": "FieldValue",
  "id": "100032",
  "value": "CDC13000000000003"
 },
 {
  "type": "FieldValue",
  "id": "100033"
 },
 {
  "type": "FieldValue",
  "id": "100034"
 },
 {
  "type": "FieldValue",
  "id": "100035"
 },
 {
  "type": "FieldValue",
  "id": "100036"
 },
 {
  "type": "FieldValue",
  "id": "100041"
 },
 {
  "type": "FieldValue",
  "id": "100043"
 },
```

```
{
  "type": "FieldValue",
  "id": "100044"
},
{
  "type": "FieldValue",
  "id": "100045"
},
{
  "type": "FieldValue",
  "id": "100046"
},
{
  "type": "FieldValue",
  "id": "100047"
},
{
  "type": "FieldValue",
  "id": "100048"
},
{
  "type": "FieldValue",
  "id": "100049"
},
{
  "type": "FieldValue",
  "id": "100051"
},
{
  "type": "FieldValue",
  "id": "100065"
},
{
  "type": "FieldValue",
  "id": "100066"
},
{
  "type": "FieldValue",
  "id": "100068"
},
```

```
{
  "type": "FieldValue",
  "id": "100069"
},
{
  "type": "FieldValue",
  "id": "100072"
},
{
  "type": "FieldValue",
  "id": "100081"
},
{
  "type": "FieldValue",
  "id": "100171",
  "value": "test.oracle.com"
},
{
  "type": "FieldValue",
  "id": "100172",
  "value": "Test User"
},
{
  "type": "FieldValue",
  "id": "100174"
},
{
  "type": "FieldValue",
  "id": "100175"
},
{
  "type": "FieldValue",
  "id": "100176"
},
{
  "type": "FieldValue",
  "id": "100177"
},
{
  "type": "FieldValue",
```

```
  "id": "100178"
 },
 {
  "type": "FieldValue",
  "id": "100179"
 },
 {
  "type": "FieldValue",
  "id": "100180"
 },
 {
  "type": "FieldValue",
  "id": "100181",
  "value": "02e0f097ee6a9f8359066e9f005c95e7"
 },
 {
  "type": "FieldValue",
  "id": "100182",
  "value": "d8f3d889e7753a6b6ad99200f8cb678b74a6f20e9fc7216725549c2eec8dc97b"
 },
 {
  "type": "FieldValue",
  "id": "100183"
 },
 {
  "type": "FieldValue",
  "id": "100184"
 },
 {
  "type": "FieldValue",
  "id": "100185"
 },
 {
  "type": "FieldValue",
  "id": "100186"
 },
 {
  "type": "FieldValue",
  "id": "100187"
 },
```

```
    {
      "type": "FieldValue",
      "id": "100188",
      "value": "TestUserL88882"
    }
   ],
   "firstName": "Test",
   "isBounceback": "false",
   "isSubscribed": "true",
   "lastName": "User",
   "subscriptionDate": "1420586365"
  }
]
```

Exporting assets

If you need to export all assets in your Eloqua database using the application API endpoints, there are URL parameters you can use to customize your requests. You can use these URL parameters in different scenarios to customize your requests and retrieve your assets.

Understanding the URL parameters

There are two URL parameters that help customize the results from the application API when retrieving lists of assets.

Query parameter	Description	Constraints
`count`	Maximum number of entities to return	Any whole number between 1 and 1000 inclusive. **Example**: Return the first page of 20 landing pages (results 1...20): `GET .../assets/landingPages?count=20`
`page`	Specifies which page of	Any positive whole number.

Query parameter	Description	Constraints
	entities to return (the count parameter defines the number of entities per page). If the page parameter is not supplied, 1 will be used by default.	**Example**: Return the second page of 20 landing pages (results 21...40): `GET .../assets/landingPages?page=2&count=20`

Example requests

When forming your requests, you may want to do an initial pull to retrieve all assets you currently have in your database, and then perform requests later on to retrieve any assets created after your initial export.

Initial export

When exporting a large number of assets, first determine the number of assets that does not result in a timeout by setting the `count` parameter. For complex Eloqua assets, such as campaigns, 10 would be a safe number, but given the complexity, and data returned at complete `depth`, setting the `count` to 5 is recommended.

Once the `count` is decided you can loop through all the pages to retrieve all the assets. To determine how many pages there are just divide `total` by `count`, and round up.

In example, using campaigns let's retrieve the first page of 10 campaigns:

```
GET /api/rest/2.0/assets/campaigns?depth=complete&count=10
```

Retrieve the second page of 10 campaigns:

```
GET /api/rest/2.0/assets/campaigns?depth=complete&count=10&page=2
```

Retrieve the third page of 10 campaigns:

```
GET /api/rest/2.0/assets/campaigns?depth=complete&count=10&page=3
```

If continuing the example above, for the last request, the value for `page` will always be the `total` number of campaigns in your database divided by the `count`, rounded up. In example, if there are 55 campaigns and you're using a count 10, the last page will be 6.

```
GET /api/rest/2.0/assets/campaigns?depth=complete&count=10&page=<total/count>
```

Alternatively, you could also keep incrementing the `page` URL parameter by 1 until you receive an empty result for the `elements` property.

Subsequent exports

After the initial export, you could update the request to retrieve only the assets which have been updated after a certain date and time. Most assets have an `updatedAt` property that could be used with the `search` URL parameter. Some assets, such as forms, specify time periods differently. This tutorial will focus on assets with `updatedAt`.

Now that you have the initial export complete, you'll only need to retrieve modified assets on a periodic basis. This export should be significantly less than the initial export, so you might not even have to specify `count`.

In example using campaigns, let's retrieve the first page of 10 campaigns that have been updated after a date and time (in this case our date and time is when the initial

export stated, and is expressed as XXXXXXXXX for a Unix timestamp in the examples below):

```
GET
/api/rest/2.0/assets/campaigns
?depth=complete&search='updatedAt>XXXXXXXXXX'&count=10
```

Retrieve the second page of 10 campaigns:

```
GET
/api/rest/2.0/assets/campaigns
?depth=complete&search='updatedAt>XXXXXXXXXX'&count=10&page=2
```

Retrieve the third page of 10 campaigns:

```
GET
/api/rest/2.0/assets/campaigns
?depth=complete&search='updatedAt>XXXXXXXXXX'&count=10&page=3
```

If continuing our example, for the last request, the value for `page` will always be the total number of entities in your database divided by the `count`, rounded up.

```
GET /api/rest/2.0/assets/campaigns
?depth=complete&search='updatedAt>XXXXXXXXXX'&count=10&page=<total/count>
```

Using the search URL parameter

The URL parameter `search` is used to customize API requests to Eloqua's application API endpoints by specifying search criteria to filter results.

The search parameter enables developers to filter for many use cases. Some examples include getting lists of assets that have been created or updated within a certain time frame, retrieving assets created by specific users, and retrieving lists of contact or account data created or updated after a specified date and time.

In this tutorial:

- Understanding how the search URL parameter works
- Examples
- Best practices for using the wildcard character - "*"
- Limit and best practices for searching custom object data
- Learn which terms are supported

Understanding how the search URL parameter works

The syntax for the search parameter is: `search={term}{operator}{value}`. Where `{term}` is the name of a field to filter on, `{operator}` is the comparison operator, and `{value}` is the value to compare the term with. Searches for exact full matches, and a * symbol can be used as a wildcard for partial matches.

You can search with fields even if they are not returned at the depth being used.

Terms

When selecting a term to search on, note that there are two categories of terms: properties and fields.

- **Properties**: Endpoint specific properties such as `updatedAt`, `createdAt`, `accessedAt`, and so on that can be found within that endpoint's documentation. Generally supported by application API */asset* endpoints. For example: `api/rest/<version>/asset/<endpoint>`.

- **Fields**: System generated fields or Custom fields created in Eloqua. Generally supported by application API */data* endpoints. For example: `api/rest/<version>/data/<endpoint>`.

Some terms support searching for blank values by including two single quotes (`?search=M_City=''`).

The endpoint you are using may support specific properties, fields, or a combination or both. See the list of terms supported with search for more information on which properties and fields are supported.

Searching using a property as a term

Here's an example using the campaigns endpoint where we are using the `updatedAt` property to search a time frame.

```
GET
/api/REST/2.0/assets/campaigns
?search=updatedAt<'1417726743'updatedAt>'1417725656'
```

> **Note**: When using `search` with a property (in example when interacting with most `/asset` endpoints) the time is expressed as a Unix time stamp for the date and time.

For most application API asset endpoints you can search with the `createdBy` and `updatedBy` properties, but the search value must be the login name, and not the user ID. For example:

```
GET /api/REST/2.0/assets/campaigns?search=createdBy=John.Doe
```

The login name is retrieved by user id with the retrieve a user or retrieve a list of users endpoints. Login name is returned with the `name` property.

Searching using a field as a term

Here's an example using the contact data endpoint where we are using the internal field name `C_DateModified` to search a time frame.

```
GET /api/REST/1.0/data/contacts?search=C_DateModified<'2018-02-06 18:50:05'C_DateModified>'2018-01-06 18:50:05'
```

Internal field names such as `C_DateModified` shown above can be retrieved using the appropriate endpoints. The table below outlines which endpoints are used to retrieve internal field names.

Entity	API	Endpoint
Accounts	Bulk API	Retrieve a list of account field definitions
Contacts	Application API	Retrieve a contact field Retrieve a list of contact fields
	Bulk API	Retrieve a list of account field definitions
Custom object records	Application API	Retrieve custom object
	Bulk API	Retrieve a list of fields for a custom object

Note: To return the internal field name for application API endpoints, `depth` must be set to `partial` or `complete`. You can search with fields even if they are not returned at the depth being used.

Searching using a field also requires the time to be expressed in one of the following formats:

- YYYY-MM-DD hh:mm:ss
- MM-DD-YYYY hh:mm:ss
- YYYY/MM/DD hh:mm:ss
- MM\/DD/YYYY hh:mm:ss

  Note:

  - The date value will need to be using Eastern Time.
  - The specificity is limited as far as seconds.
  - The time should use military time.
  - Although some date and timestamps in Eloqua may display in milliseconds, the date and time fields retrieved using the search parameter, is accurate down to seconds. For example, a last modified date of 2016-03-08 14:47:10.650 may appear for a campaign, however the search query can only request greater than 2016-03-08 14:47:10.

Operators

The following operators are supported on most endpoints:

- **=** (Equal To)
- **!=** (Not equal to)
- **>** (Greater than)
- **<** (Less than)
- **>=** (Greater than or Equal to)
- **<=** (Less than or Equal to)

If chaining more than one search criteria, each value must be placed in single quotes and will be chained as ANDs if using date fields.

```
?search=updatedAt<'1493347821'updatedAt>'1393347819'name='update*'
```

If including more than one non-date field, where they are the same field, they will be ORs.

```
?search=name='Test1'name='Test2'
```

This request would return two records, assuming they exist in Eloqua.

If including more than one different field, they will be ANDs.

```
?search=name='Test1'name='Test2'C_Company='Oracle'
```

This request will return records with a name of "Test1" or "Test2" and a Company of "Oracle".

Values

If there are spaces in the `value`, the `value` needs to be placed in single quotes:

```
?search=C_DateCreated>'2018-03-06 18:50:05'
```

Otherwise, single quotes are not required:

```
?search=updatedAt<1493347821
```

> **Note**: If there are spaces and a single quote in the value, the value needs to be placed in single quotes and the single quote in the value needs to be escaped using a single quote.

Here is example searching for an email named "Test's Test Email":

```
?search='Test''s Test Email'
```

Examples

Using the search URL parameter to search a time frame for most application API asset endpoints:

```
GET /api/REST/2.0/assets/campaigns?search=updatedAt<'1417726743'updatedAt>'1417725656'
```

Also works with `createdAt`.

Using the search URL parameter to filter results by the user who created the asset:

```
GET /api/REST/2.0/assets/campaigns?search=createdBy=John.Doe
```

Also works with `updatedBy`.

Using the search URL parameter to search a time frame for application API data endpoints:

- Contacts

  ```
  GET /api/REST/1.0/data/contacts?search=C_DateModified<'2018-02-06 18:50:05'C_DateModified>'2018-01-06 18:50:05'
  ```

  Also works with `C_DateCreated`.

- Accounts

  ```
  GET /api/REST/1.0/data/accounts?search=M_DateModified<'2018-02-06 18:50:05'M_DateModified>'2018-01-06 18:50:05'
  ```

  Also works with `M_DateCreated`.

- Custom objects

  ```
  GET /api/REST/1.0/data/customObject/9?search=ModifiedAt<'2017-06-28 00:00:00'ModifiedAt>'2017-01-09 10:15:00'
  ```

  Also works with `CreatedAt`.

Using the search URL parameter to search for blank values:

```
GET /api/REST/1.0/data/accounts?search=M_City=''
```

Using the search URL parameter to search for contacts that are linked to an account.

```
GET /api/REST/1.0/data/contacts?search=accountId>0
```

Best practices for using the wildcard character - "*"

A " * " symbol can be used for partial matches. It is recommended to use the " * " after the value whenever possible, as this produces the most efficient search.

Here are some common searches using a leading wildcard that can be changed to a more efficient search:

Endpoint	Search value	Leading wildcard search	Alternative search
/api/REST/1.0/data/contacts	Full email address	• `search=C_EmailAddress=*<fullEmailAddress>` • `search=emailAddress=*<fullEmailAddress>` • `search=common_fields=*<fullEmailAddress>`	The same results are returned by removing the leading wildcard: • `search=C_EmailAddress=<fullEmailAddress>` • `search=emailAddress=<fullEmailAddress>`

Endpoint	Search value	Leading wildcard search	Alternative search
		• `search=name=*<fullEmailAddress>` • `search=*<fullEmailAddress>`	• `search=common_fields=<fullEmailAddress>` • `search=name=<fullEmailAddress>` • `search=<fullEmailAddress>`
/api/REST/1.0/data/contacts	Email domain	• `search=C_EmailAddress=*<emailDomain>` • `search=emailAddress=*<emailDomain>` • `search=common_fields=*<emailDomain>` • `search=name=*<emailDomain>` • `search=*<emailDomain>`	The same results are returned by using the `C_EmailAddressDomain` field without a leading wildcard: • `search=C_EmailAddressDomain=<emailDomain>` The `C_EmailAddressDomain` field can also be used efficiently with a trailing wildcard: • `search=C_EmailAddressDomain=<emailDomain>*`

Limit and best practices for searching custom object data

Limit

When a Custom Object's records are over one million the Retrieve a list of custom object data Application API 1.0 and 2.0 endpoints will only accept one search request at a time that includes only non-indexed fields or does not include an indexed field without a leading wildcard per User per Custom Object. A "429 Too Many Requests" response will be returned until the search request that includes only non-indexed fields or does not include an indexed field without a leading wildcard completes.

Best practices

- **Do not use leading wildcard searches**

  It is recommended to use the " * " after the value whenever possible, as this produces the most efficient search.

- **Searching for custom object records using custom fields**

  If you plan to heavily search for custom object records by a custom field, then this custom field should be set to unique code, email, or display name field. In this case, an index will be added to the custom field.

- **Maintain a low custom object record volume**

  Keep the number of records as low as possible in custom objects. For example, you can regularly delete unneeded custom object records.

List of supported indexed fields

- 1.0
    - id
    - uniqueCode
    - ModifiedAt
    - CreatedAt
- 2.0
    - name
    - uniqueCode
    - CreatedAt

- contactId
- accountId

List of terms supported with search

> **Note**: For custom object records, you can only search by system dates using the 1.0 endpoint. The 2.0 endpoint does not support searching by system dates.

Endpoint	Properties	Fields
Accounts 1.0	• `name` • `createdAt` YYYY-MM-DD hh:mm:ss format or similar • `city` • `country`	All system and custom fields
Contact fields 1.0	• `id` • `createdAt` Unix format • `name`	
Contact lists 1.0	• `id` • `folderId` • `updatedAt` Unix format	

Endpoint	Properties	Fields
	• `createdAt` Unix format • `name`	
Contact segments 1.0	• `id` • `folderId` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `name`	
Contacts 1.0	• `id` • `createdAt` YYYY-MM-DD hh:mm:ss format or similar • `name` • `emailAddress` • `lastName` • `accountId`	All system and custom fields
Content Sections 1.0	• `id` • `folderId` • `createdBy`	

Endpoint	Properties	Fields
	• `name`	

Custom object data1.0	• `id`
	• `uniqueCode`
		• System
		  • `ModifiedAt`
		  • `CreatedAt`
		• Custom fields

Custom objects 1.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `displayNameFieldId`	
Email folders 1.0	• `id` • `name` • `createdAt` • `updatedAt` • `folderId` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Email	• `id`	

Endpoint	Properties	Fields
footers 1.0	• `name` • `folderId` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Email groups 1.0	• `id` • `name`	
Email headers 1.0	• `id` • `name` • `folderId`	
Emails 1.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Forms 1.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy`	

Endpoint	Properties	Fields
	Value must be User login name	
Images 1.0	• `id` • `name` • `createdAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Landing pages 1.0	• `id` • `name` • `createdAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Microsites 1.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy`	

Endpoint	Properties	Fields
	Value must be User login name	
Option lists 1.0	• `id` Value must be user ID • `name`	
Users 1.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Account groups 2.0	• `id` • `name` • `folderId` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Campaigns 2.0	• `id` • `name` • `createdAt` Unix format	

Endpoint	Properties	Fields
	• `updatedAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `campaignCategory` Must be specified by ID. 1 being a `contact` campaign, 2 being an `emailMarketing` campaign. • `endAt` • `crmId` • `currentStatus` Must be specified by ID: • `Active` = 1 • `Draft` = 2 • `Scheduled` = 5 • `Completed` = 7	
Contact segments	• `id`	

Endpoint	Properties	Fields
2.0	• `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
Custom object data 2.0	• `name` • `uniqueCode` • `contactId` Only the = operator • `accountId` Only the = operator	• System • `CreatedAt` • Custom fields
Custom objects 2.0	• `uniqueCode` • `name`	
Emails 2.0	• `name` • `createdAt` Unix format • `updatedAt` Unix format • `createdBy` Value must be User login name	

Endpoint	Properties	Fields

	• `updatedBy`
	
	Value must be User login name
	• `emailGroupId`
	• `updatedByUserId`
	Value must be User ID
	• `createdByUserId`
	Value must be User ID
	• `contentSource`
	Filters emails by content source ID. Specify the content source ID when searching.
	  • editor = `0`
	  • upload = `1`
	  • responsive = `3`	

Event registrants 2.0	• `name` • `uniqueCode`	
Events 2.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	

Endpoint	Properties	Fields
External asset types 2.0	• `id` • `name` • `createdBy` Value must be User login name • `updatedBy` Value must be User login name	
External assets 2.0	• `id` • `name`	
Forms 2.0	• `name` • `createdAt` Unix format • `updatedAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `updatedByUserId` Value must be User ID	

Endpoint	Properties	Fields

	• `createdByUserId`
	Value must be User ID
	• `contentSource`
	Filters forms by content source ID. Specify the content source ID when searching.
	  • editor = `0`
	  • responsive = `3`	

Form Data 2.0 (Form Spam)	• `blockreason`
	The reason for blocking the form submission. You can filter on the block reasons below:
	  • `Unknown`
	  Blocked because of an unknown reason.
	  • `InvalidToken`
	  Blocked because the form submitter had an invalid authentication token. Submitted using the form submit endpoint (a POST `formsubmissionsource`).
	  • `SiteIdMismatch`
	  Blocked because of a Site ID mismatch issue. Submitted using the form submit endpoint (a POST `formsubmissionsource`).
	  • `UserAgentMismatch`
	  Blocked because of a user agent mismatch issue. Submitted using the form submit endpoint (a POST `formsubmissionsource`).
	  • `TimestampTooFast`	

Endpoint	Properties	Fields
	Blocked because the form submit timestamp was submitted too quickly since the last form submit. Submitted using the form submit endpoint (a POST `formsubmissionsource`). • `TimestampExpired` Blocked because the timestamp expired. Submitted using the form submit endpoint (a POST `formsubmissionsource`). • `InvalidHoneypotValue` Blocked because of an invalid honey pot value. Submitted using the form submit endpoint (a POST `formsubmissionsource`). • `AuthHashMismatch` Blocked because of an authentication hash mismatch issue. Submitted using a blind form submit within an email (a GET `formsubmissionsource`). • `formsubmissionsource` • `get` Indicates the form was submitted through a blind form submit within an email. • `post` Indicates the form was submitted through a form submit endpoint.	
Landing pages 2.0	• `name` You can search a landing page's `name` and `relativePath` by using the leading wildcard "*". This	

Endpoint	Properties	Fields
	is because the `relativePath` always starts with a "/". For example: `?search=name=*My_Landing_Page` or `?search=*My_Landing_Page`. • `createdAt` Unix format • `updatedAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `updatedByUserId` Value must be User ID • `createdByUserId` Value must be User ID • `contentSource` Filters landing pages by content source ID. Specify the content source ID when searching. • editor = `0`	

Endpoint	Properties	Fields
	• upload = `1` • migration = `2` • responsive = `3`	
Programs 2.0	• `id` • `name` • `createdAt` Unix format • `updatedAt` Unix format • `createdBy` Value must be User login name • `updatedBy` Value must be User login name • `defaultEntityType` Possible values: • `Contacts` • `CustomObjectRecords` • `currentStatus`	

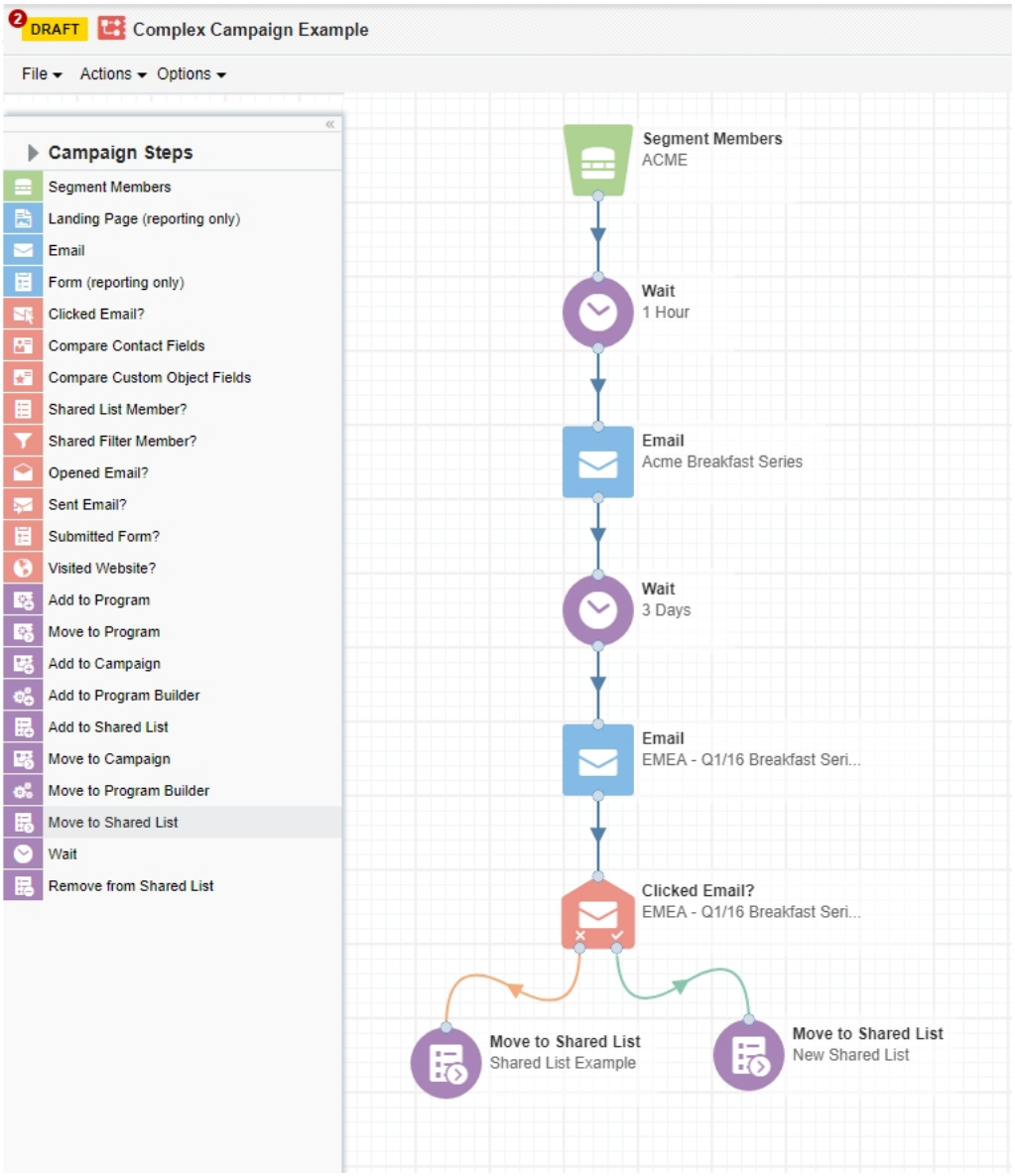
Endpoint	Properties	Fields
	Possible values: • `Draft` • `Paused` • `Active` • `defaultEntityId` For custom object programs, this is the ID of the selected custom object	

Creating campaigns with steps using the application API

There are many campaign steps available to you when creating a campaign. This tutorial outlines how to use the application API to create a campaign with campaign steps and provides an example. After you understand how to create campaign, see campaign element reference for a full list of all campaign steps.

Creating a campaign with steps

Creating a campaign with steps is done using the `elements` array. Here is an example of a campaign in Eloqua.



The following request outlines how to create this campaign using the API. Note the `elements`.

Request

```
POST /api/REST/2.0/assets/campaigns
```

Request body

```
{
  "name": "Campaign with steps example",
  "elements": [
    {
      "type": "CampaignSegment",
      "id": "-1",
      "name": "Segment Members",
      "memberCount": "0",
      "memberErrorCount": "0",
      "position": {
        "x": 167,
        "y": 53
      },
      "isFinished": "false",
      "isRecurring": "false",
      "segmentId": "38",
      "outputTerminals": [
        {
          "type": "CampaignOutputTerminal",
          "terminalType": "out",
          "connectedId": "-2",
          "connectedType": "CampaignWaitAction",
          "id": "-500011"
        }
      ],
      "reEvaluationFrequency": "3600"
    },
    {
      "type": "CampaignWaitAction",
      "name": "Wait",
      "position": {
        "x": 167,
        "y": 168
      },
      "id": "-2",
      "outputTerminals": [
        {
          "type": "CampaignOutputTerminal",
          "terminalType": "out",
          "connectedId": "-3",
```

```
      "connectedType": "CampaignEmail",
      "id": "-500012"
    }
  ],
  "waitFor": "3600"
},
{
  "type": "CampaignEmail",
  "name": "Email",
  "position": {
    "x": 167,
    "y": 281
  },
  "id": "-3",
  "includeListUnsubscribeHeader": "true",
  "outputTerminals": [
    {
      "type": "CampaignOutputTerminal",
      "terminalType": "out",
      "connectedId": "-4",
      "connectedType": "CampaignWaitAction",
      "id": "-500013"
    }
  ],
  "emailId": "61",
  "sendTimePeriod": "sendAllEmailAtOnce"
},
{
  "type": "CampaignWaitAction",
  "name": "Wait",
  "position": {
    "x": 167,
    "y": 393
  },
  "id": "-4",
  "outputTerminals": [
    {
      "type": "CampaignOutputTerminal",
      "terminalType": "out",
      "connectedId": "-5",
```

```
      "connectedType": "CampaignEmail",
      "id": "-500014"
    }
  ],
  "waitFor": "259200"
},
{
  "type": "CampaignEmail",
  "name": "Email",
  "position": {
    "x": 167,
    "y": 506
  },
  "id": "-5",
  "includeListUnsubscribeHeader": "true",
  "outputTerminals": [
    {
      "type": "CampaignOutputTerminal",
      "terminalType": "out",
      "connectedId": "-6",
      "connectedType": "CampaignEmailClickthroughRule",
      "id": "-500015"
    }
  ],
  "emailId": "52",
  "sendTimePeriod": "sendAllEmailAtOnce"
},
{
  "type": "CampaignEmailClickthroughRule",
  "name": "Clicked Email?",
  "position": {
    "x": 167,
    "y": 621
  },
  "id": "-6",
  "outputTerminals": [
    {
      "type": "CampaignOutputTerminal",
      "terminalType": "no",
      "connectedId": "-7",
```

```
      "connectedType": "CampaignMoveToContactListAction",
      "id": "-500016"
     },
     {
      "type": "CampaignOutputTerminal",
      "terminalType": "yes",
      "connectedId": "-8",
      "connectedType": "CampaignMoveToContactListAction",
      "id": "-500017"
     }
    ],
    "emailId": "52",
    "numberOfClicks": "1",
    "evaluateNoAfter": "0",
    "withinLast": "604800"
   },
   {
    "type": "CampaignMoveToContactListAction",
    "name": "Move to Shared List",
    "position": {
     "x": 52,
     "y": 735
    },
    "id": "-7",
    "listId": "96",
    "outputTerminals": [

    ]
   },
   {
    "type": "CampaignMoveToContactListAction",
    "name": "Move to Shared List",
    "position": {
     "x": 281,
     "y": 729
    },
    "id": "-8",
    "listId": "97",
    "outputTerminals": [
```

```
    ]
   }
  ]
}
```

Request notes:

- In this request body, negative numbers are used for each campaign element. This is necessary because Eloqua will set these ids when the Campaign is created; therefore, you cannot know what they are prior.
- The negative id is a placeholder that allows you to connect the steps within the output terminals. For the output terminals, the unique negative number reference id is only needed for the output terminal itself if there is more than one output terminal, i.e. a yes / no decision step.

Response


```
{
   "type": "Campaign",
   "currentStatus": "Draft",
   "id": "118",
   "createdAt": "1543427274",
   "createdBy": "9",
   "depth": "complete",
   "folderId": "1263",
   "name": "Campaign with steps example",
   "permissions": [
      "Retrieve",
      "SetSecurity",
      "Delete",
      "Update",
      "Activate"
   ],
   "updatedAt": "1543427274",
   "updatedBy": "9",
   "elements": [
```

```
{
   "type": "CampaignSegment",
   "id": "2211",
   "initialId": "-1",
   "name": "Segment Members",
   "memberCount": "0",
   "memberErrorCount": "0",
   "outputTerminals": [
      {
         "type": "CampaignOutputTerminal",
         "id": "1821",
         "initialId": "-500011",
         "connectedId": "2212",
         "connectedType": "CampaignWaitAction",
         "terminalType": "out"
      }
   ],
   "position": {
      "type": "Position",
      "x": "167",
      "y": "53"
   },
   "isFinished": "false",
   "isRecurring": "false",
   "segmentId": "38"
},
{
   "type": "CampaignWaitAction",
   "id": "2212",
   "initialId": "-2",
   "name": "Wait",
   "memberCount": "0",
   "memberErrorCount": "0",
   "outputTerminals": [
      {
         "type": "CampaignOutputTerminal",
         "id": "1822",
         "initialId": "-500012",
         "connectedId": "2213",
         "connectedType": "CampaignEmail",
```

```
            "terminalType": "out"
          }
        ],
        "position": {
          "type": "Position",
          "x": "167",
          "y": "168"
        },
        "isNotificationEnabled": "False",
        "waitFor": "3600"
      },
      {
        "type": "CampaignEmail",
        "id": "2213",
        "initialId": "-3",
        "name": "Email",
        "memberCount": "0",
        "memberErrorCount": "0",
        "outputTerminals": [
          {
            "type": "CampaignOutputTerminal",
            "id": "1823",
            "initialId": "-500013",
            "connectedId": "2214",
            "connectedType": "CampaignWaitAction",
            "terminalType": "out"
          }
        ],
        "position": {
          "type": "Position",
          "x": "167",
          "y": "281"
        },
        "sendTimePeriod": "sendAllEmailAtOnce",
        "emailId": "61",
        "includeListUnsubscribeHeader": "true",
        "isAllowingResend": "false",
        "isAllowingSentToMasterExclude": "false",
        "isAllowingSentToUnsubscribe": "false",
        "stoType": "none"
```

```
    },
    {
      "type": "CampaignWaitAction",
      "id": "2214",
      "initialId": "-4",
      "name": "Wait",
      "memberCount": "0",
      "memberErrorCount": "0",
      "outputTerminals": [
        {
          "type": "CampaignOutputTerminal",
          "id": "1824",
          "initialId": "-500014",
          "connectedId": "2215",
          "connectedType": "CampaignEmail",
          "terminalType": "out"
        }
      ],
      "position": {
        "type": "Position",
        "x": "167",
        "y": "393"
      },
      "isNotificationEnabled": "False",
      "waitFor": "259200"
    },
    {
      "type": "CampaignEmail",
      "id": "2215",
      "initialId": "-5",
      "name": "Email",
      "memberCount": "0",
      "memberErrorCount": "0",
      "outputTerminals": [
        {
          "type": "CampaignOutputTerminal",
          "id": "1825",
          "initialId": "-500015",
          "connectedId": "2216",
          "connectedType": "CampaignEmailClickthroughRule",
```

```
            "terminalType": "out"
          }
        ],
        "position": {
          "type": "Position",
          "x": "167",
          "y": "506"
        },
        "sendTimePeriod": "sendAllEmailAtOnce",
        "emailId": "52",
        "includeListUnsubscribeHeader": "true",
        "isAllowingResend": "false",
        "isAllowingSentToMasterExclude": "false",
        "isAllowingSentToUnsubscribe": "false",
        "stoType": "none"
      },
      {
        "type": "CampaignEmailClickthroughRule",
        "id": "2216",
        "initialId": "-6",
        "name": "Clicked Email?",
        "memberCount": "0",
        "memberErrorCount": "0",
        "outputTerminals": [
          {
            "type": "CampaignOutputTerminal",
            "id": "1827",
            "initialId": "-500017",
            "connectedId": "2216",
            "connectedType": "CampaignEmailClickthroughRule",
            "terminalType": "yes"
          },
          {
            "type": "CampaignOutputTerminal",
            "id": "1826",
            "initialId": "-500016",
            "connectedId": "2217",
            "connectedType": "CampaignMoveToContactListAction",
            "terminalType": "no"
          }
```

```
        ],
        "position": {
          "type": "Position",
          "x": "167",
          "y": "621"
        },
        "emailId": "52",
        "evaluateNoAfter": "0",
        "numberOfClicks": "1",
        "withinLast": "604800"
      },
      {
        "type": "CampaignMoveToContactListAction",
        "id": "2217",
        "initialId": "-7",
        "name": "Move to Shared List",
        "memberCount": "0",
        "memberErrorCount": "0",
        "position": {
          "type": "Position",
          "x": "52",
          "y": "735"
        },
        "listId": "96"
      },
      {
        "type": "CampaignMoveToContactListAction",
        "id": "2218",
        "initialId": "-8",
        "name": "Move to Shared List",
        "memberCount": "0",
        "memberErrorCount": "0",
        "position": {
          "type": "Position",
          "x": "281",
          "y": "729"
        },
        "listId": "97"
      }
    ],
```

```
    "isReadOnly": "false",
    "actualCost": "0",
    "budgetedCost": "0",
    "campaignCategory": "contact",
    "campaignType": "",
    "crmId": "",
    "fieldValues": [
        {
            "type": "FieldValue",
            "id": "9",
            "value": ""
        },
        {
            "type": "FieldValue",
            "id": "10",
            "value": ""
        },
        {
            "type": "FieldValue",
            "id": "11",
            "value": ""
        }
    ],
    "isEmailMarketingCampaign": "false",
    "isIncludedInROI": "false",
    "isMemberAllowedReEntry": "false",
    "isSyncedWithCRM": "false",
    "product": "",
    "region": ""
}
```

Check out campaign element reference to learn how to create campaigns with other campaign elements.

Mapping contacts via form submit

When you submit form data via API, you can map these contacts in your Eloqua instance. This tutorial will walkthrough the required form configuration needed to map

contacts.

In this tutorial:

Step 1 - Create the form and configure the form fields

Create the form using the API endpoints (example provided below) or the Eloqua interface.

Your form must contain at least two fields:

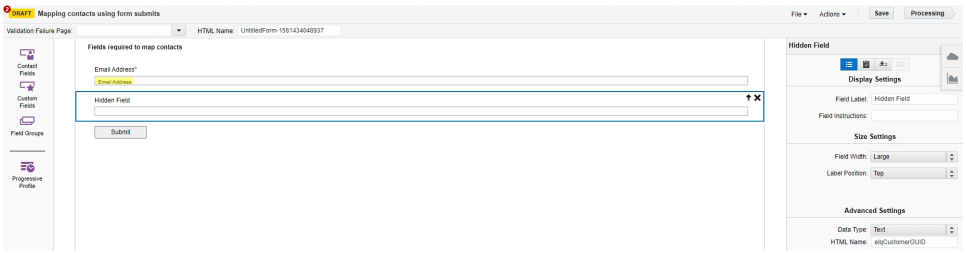
- **Email Address**

  For the HTML name, specify a name of your choice.

- **Hidden Field**

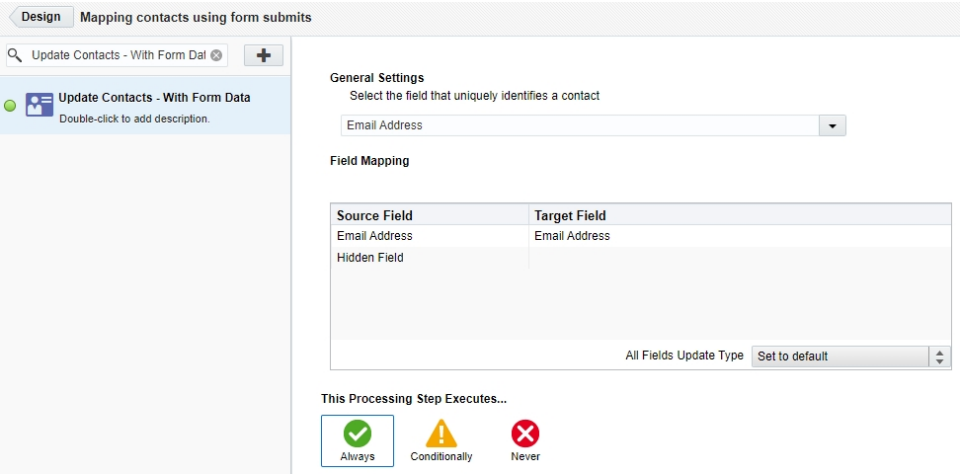
  The Hidden Field is used to store the customer's GUID. For the HTML name, specify `elqCustomerGUID`.

Here's an example form field configuration shown in the user interface.



Step 2 - Configure the form processing steps

Your form must contain the Form Processing Step **Update Contacts - With Form Data**.



Creating the form using the API

Here's an example of how to create this form using the application 2.0 API endpoint.

Request URL

```
POST  /api/REST/2.0/assets/form
```

Request Body

```
{
 "name": "Mapping contacts via form submit",
 "elements": [
  {
   "name": "Email Address",
   "htmlName": "emailAddress",
   "dataType": "text",
   "displayType": "text",
   "defaultValue": "",
   "type": "FormField",
   "style": "{\"fieldSize\":\"large\"}",
   "fieldMergeId": "1",
   "id": "-1",
   "validations": [
    {
     "id": "-2",
```

```
      "message": "This field is required",
      "type": "FieldValidation",
      "condition": {
       "type": "IsRequiredCondition"
      }
     },
     {
      "id": "-3",
      "message": "A valid email address is required",
      "type": "FieldValidation",
      "condition": {
       "type": "IsEmailAddressCondition"
      }
     }
    ]
   },
   {
    "name": "elqCustomerGUID",
    "htmlName": "elqCustomerGUID",
    "type": "FormField",
    "style": "{\"fieldSize\":\"large\"}",
    "dataType": "text",
    "displayType": "hidden",
    "id": "-4",
    "validations": [
     {
      "id": "-5",
      "message": "Value must not contain any URL's",
      "type": "FieldValidation",
      "condition": {
       "type": "PreventUrlCondition"
      }
     },
     {
      "id": "-6",
      "message": "Value must not contain any HTML",
      "type": "FieldValidation",
      "condition": {
       "type": "PreventXSSCondition"
      }
```

```
      },
      {
        "type": "FieldValidation",
        "id": "-7",
        "message": "Invalid length for field value",
        "condition": {
          "maximum": "35",
          "minimum": "0",
          "type": "TextLengthCondition"
        }
      }
    ]
  },
  {
    "name": "Submit",
    "htmlName": "submit",
    "fieldType": "submit",
    "type": "FormField",
    "style": "{\"fieldSize\":\"large\"}",
    "displayType": "submit",
    "style": "{\"fieldSize\":\"large\",\"submitButtonStyleType\":\"standard\"}",
    "id": "-8",
    "validations": [

    ],
    "hasProbableError": false,
    "instructions": ""
  }
],
"processingSteps": [
  {
    "type": "FormStepCreateUpdateContactFromFormField",
    "execute": "always",
    "id": "-9",
    "mappings": [
      {
        "type": "FormFieldUpdateMapping",
        "sourceFormFieldId": "-1",
        "targetEntityFieldId": "100001",
        "updateType": "useFieldRule",
```

```
        "id": "-10"
      },
      {
        "type": "FormFieldUpdateMapping",
        "sourceFormFieldId": "-4",
        "updateType": "useFieldRule",
        "id": "-11"
      }
    ],
    "keyFieldId": "100001",
    "hasValidationIssue": "false"
  }
 ],
 "style": "{\"fieldSize\":\"medium\",\"labelPosition\":\"top\"}",
 "processingType": "externalWebsite"
}
```

Considerations

- The **Email Address** field must be mapped, but the **GUID** hidden field does not need to be mapped. Additional fields can be mapped as appropriate.
- The field to store the **GUID** must have the exact HTML name specified above (`elqCustomerGUID`).
- The **GUID** hidden field value must be in GUID format "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx". The dashes are not required. To submit form data when you do not have a GUID value, use a value with all zeros for the field. For example: "00000000-0000-0000-0000-000000000000" or "00000000000000000000000000000000".
- If the Email Address doesn't exist, a new Contact would be created and linked to the visitor; otherwise, the Contact with the matching Email Address will be linked to the visitor indicated.
- When form data is reposted to Oracle Eloqua servers from a server-side form processor,

`elqCookieWrite` should be specified and set to 0. More information about reposting can be found in the Eloqua Help Center.

Using multiple branded domains with the application API

When Multiple Branded Domains is enabled in your instance, a new property named `brandId` is added to the Application API Email endpoints.

> **Important**: To access this feature, please contact your account representative to purchase the Oracle Eloqua Premium Branding and Configuration Cloud Service - Multiple Brands.

In this topic:

When Multiple Branded Domains is enabled, the following endpoints will be impacted:

- 1.0
  - Retrieve a list of emails: GET /api/REST/1.0/assets/emails

    Default depth is minimal
  - Retrieve an email: GET /api/REST/1.0/assets/email/ {id}

    Default depth is complete
  - Create an email: POST /api/REST/1.0/assets/email

    Default depth is complete

    - Update an email: PUT /api/REST/1.0/assets/email/{id}

      Default depth is complete

- 2.0

    - Retrieve a list of emails: GET /api/REST/2.0/assets/emails

      Default depth is minimal

    - Retrieve an email: GET /api/REST/2.0/assets/email/ {id}

      Default depth is complete

    - Create an email: POST /api/REST/2.0/assets/email

      Default depth is complete

    - Update an email: PUT /api/REST/2.0/assets/email/{id}

      Default depth is complete

brandId notes

- `brandId` is returned at complete and partial depth, when retrieving, updating, or creating an email.
- `brandId` is returned as a string that indicates the ID of the selected Brand.
- When creating a new email in Eloqua, `brandId` is set to the default Brand.
- Emails created prior to enabling this feature will not return `brandId` until the email is updated.
- When creating an email using the API, if `brandId` is not included in the request, `brandId` will be set to the default Brand.
- When updating an email using the API, if `brandId` is not included in the request, `brandId` will not be modified if set.

- To retrieve the Brand IDs, create emails in the UI set to each Brand, then retrieve each email using the Retrieve an email endpoint to view the ID.

Example requests

Create a new email without specifying brandId

Request:

```
POST /api/REST/2.0/assets/email
{
   "name": "multiple_branded_domains_API_create_default",
   "subject": "Test subject line",
   "htmlContent": {
      "type": "RawHtmlContent",
      "html": "<html><head></head><body>This is the email.</body></html>"
   }
}
```

Response:

```
{
   "type": "Email",
   "currentStatus": "Draft",
   "id": "190980",
   "createdAt": "1622058063",
   "createdBy": "3",
   "depth": "complete",
   "folderId": "42",
   "name": "multiple_branded_domains_API_create_default",
   "permissions": [
      "Retrieve",
      "SetSecurity",
      "Delete",
      "Update"
   ],
   "updatedAt": "1622058063",
   "updatedBy": "3",
```

```
    "archived": "false",
    "bounceBackEmail": "newclient@test.com",
    "brandId": "43",
    "contentSections": [],
    "dynamicContents": [],
    "encodingId": "3",
    "fieldMerges": [],
    "forms": [],
    "htmlContent": {
        "type": "RawHtmlContent",
        "contentSource": "upload",
        "html": "<html><head></head><body>This is the email.</body></html>"
    },
    "hyperlinks": [],
    "images": [],
    "isContentProtected": "false",
    "isPlainTextEditable": "false",
    "isPrivate": "False",
    "isTracked": "false",
    "layout": "",
    "plainText": "This is the email.",
    "renderMode": "Fixed",
    "replyToEmail": "newclient@test.com",
    "replyToName": "Test",
    "sendPlainTextOnly": "false",
    "senderEmail": "newclient@test.com",
    "senderName": "Test",
    "style": "",
    "subject": "Test subject line",
    "virtualMTAId": "1"
}
```

Create a new email setting brandId to 140

Request:

```
POST /api/REST/2.0/assets/email
{
    "name": "multiple_branded_domains_API_create_140",
    "subject": "Test subject line",
```

```
  "htmlContent": {
    "type": "RawHtmlContent",
    "html": "<html><head></head><body>This is the email.</body></html>"
  },
  "brandId": "140"
}
```

Response:

```
{
  "type": "Email",
  "currentStatus": "Draft",
  "id": "190981",
  "createdAt": "1622058361",
  "createdBy": "3",
  "depth": "complete",
  "folderId": "42",
  "name": "multiple_branded_domains_API_create_140",
  "permissions": [
    "Retrieve",
    "SetSecurity",
    "Delete",
    "Update"
  ],
  "updatedAt": "1622058361",
  "updatedBy": "3",
  "archived": "false",
  "bounceBackEmail": "newclient@test.com",
  "brandId": "140",
  "contentSections": [],
  "dynamicContents": [],
  "encodingId": "3",
  "fieldMerges": [],
  "forms": [],
  "htmlContent": {
    "type": "RawHtmlContent",
    "contentSource": "upload",
    "html": "<html><head></head><body>This is the email.</body></html>"
  },
```

```
    "hyperlinks": [],
    "images": [],
    "isContentProtected": "false",
    "isPlainTextEditable": "false",
    "isPrivate": "False",
    "isTracked": "false",
    "layout": "",
    "plainText": "This is the email.",
    "renderMode": "Fixed",
    "replyToEmail": "newclient@test.com",
    "replyToName": "Test",
    "sendPlainTextOnly": "false",
    "senderEmail": "newclient@test.com",
    "senderName": "Test",
    "style": "",
    "subject": "Test subject line",
    "virtualMTAId": "1"
}
```

Update the previously created email to brandId 43

Request:

```
PUT /api/REST/2.0/assets/email/190981
{
    "name": "multiple_branded_domains_API_create_secondary_updateTo43",
    "id": "190981",
    "brandId": "43"
}
```

Response:

```
{
    "type": "Email",
    "currentStatus": "Draft",
    "id": "190981",
    "initialId": "190981",
    "createdAt": "1622058361",
    "createdBy": "3",
```

```
"depth": "complete",
"folderId": "42",
"name": "multiple_branded_domains_API_create_secondary_updateTo43",
"permissions": [
   "Retrieve",
   "SetSecurity",
   "Delete",
   "Update"
],
"updatedAt": "1622058613",
"updatedBy": "3",
"archived": "false",
"bounceBackEmail": "newclient@test.com",
"brandId": "43",
"contentSections": [],
"dynamicContents": [],
"encodingId": "3",
"fieldMerges": [],
"forms": [],
"htmlContent": {
   "type": "RawHtmlContent",
   "contentSource": "upload",
   "html": "<html><head></head></html>"
},
"hyperlinks": [],
"images": [],
"isContentProtected": "false",
"isPlainTextEditable": "false",
"isPrivate": "False",
"isTracked": "false",
"layout": "",
"plainText": " ",
"renderMode": "Fixed",
"replyToEmail": "newclient@test.com",
"replyToName": "Test",
"sendPlainTextOnly": "false",
"senderEmail": "newclient@test.com",
"senderName": "Test",
"style": "",
"subject": "Test subject line",
```

```
  "virtualMTAId": "1"
}
```

Using form spam protection with the Application API

When Form Spam Protection is enabled, a new property `isFormSpamProtectionEnabled`, is added to the Application API 2.0 Form endpoints. This topic explains how this feature changes working with the Application API endpoints.

> **Note**: This feature is currently released under our Controlled Availability program. To request access to this feature, please contact your account representative for more information.

In this topic:

Impacted Endpoints

When Form Spam Protection is enabled, the following endpoints are impacted:

2.0

- Retrieve a list of forms: GET /api/REST/2.0/assets/forms

  Default depth is minimal.

- Retrieve a form: GET /api/REST/2.0/assets/form/ {id}

Default depth is complete.

- Create a form: POST /api/REST/2.0/assets/form

  Default depth is complete.

- Update a form: PUT /api/REST/2.0/assets/form/{id}

  Default depth is complete.

- Partially update a form: PATCH /api/REST/2.0/assets/form/{id}

  Default depth is complete.

isFormSpamProtectionEnabled Notes

- `isFormSpamProtectionEnabled` is returned at complete depth, when retrieving, updating, or creating a form.
- `isFormSpamProtectionEnabled` is returned as a string with the only two possible values being "true" or "false".
- When creating a new form in Eloqua `isFormSpamProtectionEnabled` is set to "false" by default.
- Forms created prior to enabling this feature will have `isFormSpamProtectionEnabled` set to "false".
- When creating a form using the API, if `isFormSpamProtectionEnabled` is not included in the request, `isFormSpamProtectionEnabled` will be set to "false".
- **When updating a form using the API**, if `isFormSpamProtectionEnabled` is not included in the request, `isFormSpamProtectionEnabled` will be set to "false".

- When creating a form using the API, if `isFormSpamProtectionEnabled` is set to "true", or updating/patching a form if changing `isFormSpamProtectionEnabled` from "false" to "true" `honeypotHtmlName` must be included in the request, and if generating custom form html, then the API user must include `honeypotHtmlName` in the html.
    - User must have Change Form Spam Protection Status action permission in order to set `isFormSpamProtectionEnabled` to "true". Learn more about setting the action permissions for a security group. Learn more about Enabling spam protection for forms.
    - If you'd like Eloqua to generate the html, open the form in Eloqua and click Save.
- The Application API 1.0 Form endpoints will not return `isFormSpamProtectionEnabled`, and if included in a Application API 1.0 Form request it will be ignored.

Example requests

Creating new form without specifying isFormSpamProtectionEnabled

Request:

```
POST /api/REST/2.0/assets/form
{
 "name": "isFormSpamProtectionEnabled not included",
 "elements": [
  {
   "type": "FormField",
   "name": "Favorite color.",
   "style": "{\"fieldSize\":\"large\",\"labelPosition\":\"top\"}",
   "dataType": "text",
   "displayType": "text",
   "htmlName": "favCol"
  }
 ],
 "processingType": "externalEmail"
}
```

Response:

201 Created


```
{
   "type": "Form",
   "currentStatus": "Draft",
   "id": "84",
   "createdAt": "1639173843",
   "createdBy": "11",
   "depth": "complete",
   "folderId": "7",
   "name": "isFormSpamProtectionEnabled not included",
   "permissions": [
      "Retrieve",
      "SetSecurity",
      "Delete",
      "Update"
   ],
   "updatedAt": "1639173843",
   "updatedBy": "11",
   "archived": "false",
   "elements": [
      {
         "type": "FormField",
         "id": "419",
         "name": "Favorite color.",
         "style": "{\"fieldSize\":\"large\",\"labelPosition\":\"top\"}",
         "dataType": "text",
         "displayType": "text",
         "htmlName": "favCol",
         "useGlobalSubscriptionStatus": "False",
         "validations": []
      }
   ],
   "formJson": "{\"type\":\"responsiveForm\",\"version\":\"1\"}",
   "isFormSpamProtectionEnabled": "false",
   "isHidden": "false",
   "isResponsive": "true",
   "processingSteps": [],
   "processingType": "externalEmail"
}
```

Creating new form setting isFormSpamProtectionEnabled to "true" requires also setting honeypotHtmlName

Request:

```
POST /api/REST/2.0/assets/form
{
  "name": "isFormSpamProtectionEnabled true",
  "elements": [
    {
      "type": "FormField",
      "name": "Favorite color.",
      "style": "{\"fieldSize\":\"large\",\"labelPosition\":\"top\"}",
      "dataType": "text",
      "displayType": "text",
      "htmlName": "favCol"
    }
  ],
  "processingType": "externalEmail",
  "isFormSpamProtectionEnabled": "true"
  "honeypotHtmlName": "address5"
}
```

Response:

```
{
    "type": "Form",
    "currentStatus": "Draft",
    "id": "85",
    "createdAt": "1639174129",
    "createdBy": "11",
    "depth": "complete",
    "folderId": "7",
    "name": "isFormSpamProtectionEnabled true",
    "permissions": [
        "Retrieve",
        "SetSecurity",
        "Delete",
        "Update"
```

```
  ],
  "updatedAt": "1639174129",
  "updatedBy": "11",
  "archived": "false",
  "elements": [
    {
      "type": "FormField",
      "id": "420",
      "name": "Favorite color.",
      "style": "{\"fieldSize\":\"large\",\"labelPosition\":\"top\"}",
      "dataType": "text",
      "displayType": "text",
      "htmlName": "favCol",
      "useGlobalSubscriptionStatus": "False",
      "validations": []
    }
  ],
  "formJson": "{\"type\":\"responsiveForm\",\"version\":\"1\"}",
  "honeypotHtmlName": "address5",
  "isFormSpamProtectionEnabled": "true",
  "isHidden": "false",
  "isResponsive": "true",
  "processingSteps": [],
  "processingType": "externalEmail"
}
```


Activity detail values

- Version 1.0
- Version 2.0

Version 1.0

The table below lists all contact activity detail values retrieved using the GET /api/REST/1.0/data/activities/contact/{id} endpoint.

Example request

Retrieve all `campaignMembership` activities associated with contact id #1 in the given time frame:

```
GET
api/rest/1.0/data/activities/contact/1?type=campaignMembership&startDate
=1262304000&endDate=1531924526
```

Request response:


```
[
 {
  "type": "CampaignActivity",
  "activityDate": "1469719163",
  "activityType": "campaignMembership",
  "asset": "54",
  "assetType": "campaign",
  "contact": "1",
  "details": [
   {
    "Key": "Responded",
    "Value": "Yes"
   },
   {
    "Key": "CampaignName",
    "Value": "EMEA Q1/16 Breakfast Roadshow"
   },
   {
    "Key": "LeadStage",
    "Value": "Prospect"
   },
   {
    "Key": "ActivityType",
    "Value": "EmailsSent"
   },
   {
    "Key": "AssetType",
    "Value": "Emails"
   },
```

```
    {
      "Key": "AssetId",
      "Value": "103"
    },
    {
      "Key": "CampaignId",
      "Value": "54"
    }
   ],
   "id": "54",
   "hasResponded": "true"
  }
]
```

Asset type	Activity type	Detail Name	Description
campaign	campaignMembership	Responded	Whether the contact has entered a campaign and met a response rule
		CampaignName	The campaign asset's name
		LeadStage	The contact's current stage in the marketing funnel
email	emailClickThrough	EmailClickedThruLink	The URL within the email the contact clicked
		EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment

Asset type	Activity type	Detail Name	Description
	emailOpen	EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		IPAddress	The IP Address where the email was opened
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment
	emailSend	EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment
	emailSubscribe	CampaignName	The campaign asset's name
		EmailCampaignId	The ID of the email group
	emailUnsubscribe	CampaignName	The campaign asset's name
		EmailCampaignId	The ID of the email group
form	formSubmit	Collection	A collection of form field data submitted
		FormName	The form asset's name
web	webVisit	Duration	Length of time spent on the web visit
		QueryString	Query string of the visited URL.
		QueryStringDisplay	Display name of the query string parameter
		URL	The URL visited
		Thread	The thread ID of the web visit

The table below lists all contact activity detail values retrieved using the

GET/api/REST/2.0/data/activities/contact/{id} endpoint.

Asset type	Activity type	Detail Name	Description
campaign	campaignMembership	Responded	Whether the contact has entered a campaign and met a response rule
		CampaignName	The campaign asset's name
		LeadStage	The contact's current stage in the marketing funnel
email	emailClickThrough	EmailClickedThruLink	The URL within the email the contact clicked
		EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment
	emailOpen	EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		IPAddress	The IP Address where the email was opened
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment

Asset type	Activity type	Detail Name	Description
	emailSend	EmailName	The email asset's name
		EmailWebLink	The URL to the online version of the email
		EmailRecipientId	The ID of the email recipient
		SubjectLine	The email asset's subject line
		DeploymentId	The ID associated with the email deployment
	emailSubscribe	CampaignName	The campaign asset's name
		EmailCampaignId	The ID of the email group
	emailUnsubscribe	CampaignName	The campaign asset's name
		EmailCampaignId	The ID of the email group
form	formSubmit	Collection	A collection of form field data submitted
		FormName	The form asset's name
web	webVisit	Duration	Length of time spent on the web visit
		QueryString	Query string of the visited URL.
		QueryStringDisplay	Display name of the query string parameter
		URL	The URL visited
		Thread	The thread ID of the web visit
sms	smsSend	Purpose	The SMS asset’s purpose (Transactional/Promotional)
		SmsCode	Sender codes
	smsDelivery	Purpose	The SMS asset’s purpose (Transactional/Promotiona

Asset type	Activity type	Detail Name	Description
			l)

Asset type	Activity type	Detail Name	Description
		SmsCode	Sender codes
		SmsDeliveredId	SMS delivery ID
	smsClickThrough	Purpose	The SMS asset's purpose (Transactional/Promotional)
		SmsClickedThroughLink	Clicked URL
	smsReply	Purpose	The SMS asset's purpose (Transactional/Promotional)
		SmsCode	Sender codes
		Keyword	SMS keyword
	smsBounceback	Purpose	The SMS asset's purpose (Transactional/Promotional)
		DRStatusCodeDescription	Description of the delivery receipt, from carrier
		Category	Type of bounceback (Soft/Hard)
	smsFail	SmsCode	Sender codes
		Error	Error description
	smsSubscribe	NA	
	smsUnsubscribe	NA	

Campaign element reference

This topic outlines the JSON objects needed to create each campaign step when creating campaigns. Insert these objects into the `elements` array when creating campaigns using the application API.

Element	Example

Segment Members	[
	{
	"type": "CampaignSegment",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Segment Members",
	"memberCount": "0",
	"memberErrorCount": "0",
	"outputTerminals": [
	{
	"type": "CampaignOutputTerminal",
	"id": "1814",
	"connectedId": "2204",
	"connectedType": "CampaignWaitAction",
	"terminalType": "out"
	}
	]
	}
	]

Landing Page (reporting only)	[
	{
	"type": "CampaignLandingPage",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Landing Page (reporting only)",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "116",
	"y": "82"
	}

Element	Example
	<pre> }]</pre>

Email	<pre>[
	 {
	  "type": "CampaignEmail",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Email",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "120",
	  "y": "83"
	  },
	  "sendTimePeriod": "sendAllEmailAtOnce",
	  "includeListUnsubscribeHeader": "true",
	  "isAllowingResend": "false",
	  "isAllowingSentToMasterExclude": "false",
	  "isAllowingSentToUnsubscribe": "false",
	  "stoType": "none"
	 }
	]</pre>

Form (reporting only)	<pre>[
	 {
	  "type": "CampaignForm",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Form (reporting only)",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "0",
	  "y": "52"
	  }
	 }</pre>

Element	Example
	]

Clicked Email?	[
	  {
	  "type": "CampaignEmailClickthroughRule",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Clicked Email?",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "0",
	  "y": "51"
	  },
	  "evaluateNoAfter": "0",
	  "numberOfClicks": "1",
	  "withinLast": "604800"
	  }
	]

Compare Contact Fields	[
	  {
	  "type": "CampaignContactFieldComparisonRule",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Compare Contact Fields",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "0",
	  "y": "50"
	  },
	  "evaluateNoAfter": "0"
	  }
	]

Compare Custom Object Fields	[

Element	Example

	{
	"type": "CampaignCustomObjectFieldComparisonRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Compare Custom Object Fields",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "57"
	},
	"evaluateNoAfter": "0"
	}
	]

Shared List Member?	[
	{
	"type": "CampaignContactListMembershipRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Shared List Member?",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "1",
	"y": "55"
	},
	"evaluateNoAfter": "0"
	}
	]

Shared Filter Member?	[{ "type": "CampaignContactFilterMembershipRule", "id": "<nextAvailableNegativeNumber>", "name": "Shared Filter Member?", "memberCount": "0",

Element	Example

	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "73"
	},
	"evaluateNoAfter": "0"
	}
	]

Opened Email?	[
	{
	"type": "CampaignEmailOpenedRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Opened Email?",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "56"
	},
	"evaluateNoAfter": "0",
	"numberOfOpens": "1",
	"withinLast": "604800"
	}
	]

Sent Email?	[
	{
	"type": "CampaignEmailSentRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Sent Email?",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",

Element	Example

	<pre>"x": "0",
	"y": "63"
	},
	"evaluateNoAfter": "0"
	}
	]</pre>

Submitted Form?	<pre>[
	{
	"type": "CampaignSubmitFormRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Submitted Form?",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "39"
	},
	"evaluateNoAfter": "0",
	"withinLast": "604800"
	}
	]</pre>

Visited Website?	<pre>[
	{
	"type": "CampaignWebsiteVisitRule",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Visited Website?",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "40"
	},
	"evaluateNoAfter": "0",</pre>

Element	Example
	<pre> "numberOfVisits": "1", "withinLast": "604800" }]</pre>

Add to Program	<pre>[
	 {
	  "type": "CanvasAddToProgramAction",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Add to Program",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "0",
	  "y": "45"
	  }
	 }
	]</pre>

Move to Program	<pre>[
	 {
	  "type": "CanvasMoveToProgramAction",
	  "id": "<nextAvailableNegativeNumber>",
	  "name": "Move to Program",
	  "memberCount": "0",
	  "memberErrorCount": "0",
	  "position": {
	  "type": "Position",
	  "x": "0",
	  "y": "51"
	  }
	 }
	]</pre>

Add to Campaign	<pre>[
	 {</pre>

Element	Example

	"type": "CampaignAddToCampaignAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Add to Campaign",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "6",
	"y": "70"
	}
	}
	]

Add to Program Builder	[
	{
	"type": "CampaignAddToProgramBuilderAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Add to Program Builder",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "41"
	}
	}
	]

Add to Shared List	[
	{
	"type": "CampaignAddToContactListAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Add to Shared List",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",

Element	Example
	"x": "0", "y": "47" } }]

Move to Campaign	[
	{
	"type": "CampaignMoveToCampaignAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Move to Campaign",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "51"
	}
	}
	]

Move to Program Builder	[
	{
	"type": "CampaignMoveToProgramBuilderAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Move to Program Builder",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "41"
	}
	}
	]

Move to Shared List	[

Element	Example

	{
	"type": "CampaignMoveToContactListAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Move to Shared List",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "26"
	}
	}
	]

Wait	[
	{
	"type": "CampaignWaitAction",
	"id": "<nextAvailableNegativeNumber>",
	"name": "Wait",
	"memberCount": "0",
	"memberErrorCount": "0",
	"position": {
	"type": "Position",
	"x": "0",
	"y": "76"
	},
	"isNotificationEnabled": "False"
	}
	]

Remove from Shared List	[{ "type": "CampaignRemoveFromContactListAction", "id": "<nextAvailableNegativeNumber>", "name": "Remove from Shared List", "memberCount": "0", "memberErrorCount": "0",

Element	Example

	"position": {
	"type": "Position",
	"x": "0",
	"y": "57"
	}
	}
	]

Oracle Eloqua

Developer Guide

Contents

Oracle Eloqua Bulk API

The Bulk API is a RESTful API designed to support high volume data transfers. It can be used for CRM and data warehousing integrations and extending Eloqua's functionality. This section includes the following topics

- **Getting Started**: This section discusses core concepts related to Eloqua's Bulk API: the API call format, descriptions of the assets and entities you will be working with, and provides some troubleshooting guidance.
    - Bulk API getting started overview
    - Learn about basic call format
    - Learn about **Eloqua elements**
- **Tutorials**: This section provides step-by-step instructions for common tasks you may want to accomplish with the bulk API such as importing and exporting data and searching for fields.
    - Exporting data from Eloqua
    - Filtering data to export
    - Importing data into Eloqua
    - Using import and export parameters
    - Sourcing **Eloqua element** metadata (to structure export and import definitions)
    - Filtering
    - Retrieving large volumes of data
    - Exporting activities
        - Activities export definitions
    - Using the campaign response endpoints
    - Retrieving app records using the bulk API
    - Uploading file data using cURL

- **Best Practices**: This section details best practices to follow when using the Bulk API.
    - Bulk API Best Practices
- **Endpoints**: This section provides detailed information about the API's service endpoints and their parameters.
    - API reference
- **Reference**: This section provides detailed reference materials for the API itself. This section includes material on:
    - Export characteristics
    - Import parameters:
        - Import characteristics
        - Import update rule types
    - Syncs:
        - Sync actions
        - Sync status types
    - Activity fields
    - Bulk data limits
    - Field names
    - Bulk API data types
    - Bulk languages:
        - Eloqua expression language
        - Eloqua markup language v2
        - Eloqua markup language v3
    - Troubleshooting

Getting started with the bulk API

While the bulk API is designed for ease of use, there are things to bear in mind when determining how you will interact with the bulk API.

Requirements

The bulk API is designed to let developers start developing with minimal setup or configuration effort. At a minimum, you need an Eloqua instance and an account on that instance. In order to be able to interact with your Eloqua data, your account needs adequate permissions to access contact fields, secondary assets, and so on.

Some Eloqua instances enable contact-level security, which restricts access to data based on different user roles. Users might only have access to contacts located in their geographical region, for instance. Because these permissions affect what data the user can access, it is important that the Eloqua user accessing the API has the appropriate permissions.

Considerations

Use case

The bulk API does not constrain how you interact with it, but your use case should determine how you develop your integration.

For example, if you are importing data into Eloqua, you should consider *resiliency*: if an item fails to import, can you send it again in 15 minutes or do you need to deal with the failure immediately? Your use case will determine what measures you need to put into place to deal with this.

Similarly, if you are exporting data from Eloqua, how large a data set are you working with? Larger exports will be time consuming, so you can break the export up into smaller chunks using a filter. Be aware of the data volumes you expect to deal with and create your export definitions accordingly.

The Troubleshooting section discusses ways you can monitor your imports and exports and leverage this data to improve your integration. The bulk API Limits documentation can also help guide you when designing your interaction with the bulk API.

Authentication and security

Access to the Oracle Eloqua bulk API requires adequate permissions based on security group. Oracle Eloqua utilizes SSL/TLS with support for 128-bit and 256-bit ciphers to securely transmit traffic in all API calls. The encryption level depends on the libraries being used by your app.

For authentication, Eloqua's bulk API supports HTTP basic authentication as well as OAuth 2.0.

OAuth is the preferred method of authenticating. OAuth allows bulk API-powered applications to access resources on behalf of a resource owner without needing the resource owner’s credentials.

The bulk API supports 2-legged OAuth for second party apps, and 3-legged OAuth for apps in the AppCloud™.

In cases where implementing OAuth is not feasible, you can use HTTP basic authentication. This approach requires that you obtain the user's credentials, which is

not ideal. For example, if a user changes their password, you need to obtain the new password. If feasible, use OAuth over basic authentication.

Accessible elements

The bulk API is gives you access to contacts, accounts, custom data objects, email groups, external activities, events, campaign responses, and opportunities. See: Introduction to Eloqua Elements for a full description of each element.

API call format

Interactions with the bulk API follow a consistent pattern: you create a record definition that tells Eloqua what data you are importing or exporting, move that data into a staging area, and either retrieve the data, in the case of an export, or push that data into Eloqua, in the case of an import.

The API call format topic explores this pattern in detail, touching on the rationale behind the three-step workflow. It also discusses data formatting requirements, header requirements, and so on.

You should familiarize yourself with the API call format before starting to work with the bulk API.

API call format

All bulk API calls use the same general call pattern: whether you are importing or exporting data, the structure of your call will be consistent.

Pattern

Importing and exporting data with the bulk API follows a three-step pattern:

1. Define the import or export. The definition describes how Eloqua's fields map to your own. For example, your email field might map to Eloqua's `Contact.Field(C_EmailAddress)`. You can also give the definition a name, so that you can reuse it later, specify how to filter the data, and define any additional actions you want to perform. The import or export definition tells Eloqua what kind of operation to prepare to perform.
2. Move data into the staging area. For imports, this means `POST`ing your data to the staging area; for exports, this means telling Eloqua to move its data into the staging area. The staging area is a middle ground that allows read and write operations to occur asynchronously from the HTTP requests you send.
3. Move the data to its final destination. For imports, this means telling Eloqua to sync the data from the staging area into its database; for exports, this involves retrieving the data from the staging area.

This design allows for fast client requests, while longer actions are performed asynchronously. By not interacting with the Eloqua database during the HTTP request, you gain a consistent expectation of how long I/O operations will take. The asynchronous pattern also enables scalability: were I/O operations and merges performed inline, database locks would impact the performance of your Eloqua instance.

The staging area system allows data to be written to disk, and then processed in the background without affecting performance.

Basic structure

Familiarizing yourself with the common URI parameters, required HTTP headers, and JSON patterns will give you a strong foundation from which to start using the bulk API.

Call URL

The URL you need to call to access Eloqua's Bulk API depends on which "pod" your Eloqua instance is hosted on. The base url is:

`https://<host>.eloqua.com/api/bulk/2.0`, where `<host>` can be `secure`, `www02.secure`, or `secure.p03`. To find your URL, see: Determining base URLs .

URL parameters

The bulk API's HTTP `GET` endpoints support a variety of URL parameters that you can use to control what data you retrieve.

- `limit`: Specifies the maximum number of records to return
- `offset`: Specifies an offset that allows you to retrieve the next batch of records. For example, if your `limit` is 1000, specifying an `offset` of 1000 will return records 1000 through 2000.
- `q`: Specifies query criteria used to filter results. The `q` format is `<term><operator><value>`. For example: `name='Email*''`.
- `orderBy`: Specifies the name of the property to order the results by. The `orderBy` format is `<term>` `ASC` | `DESC`. For example, `orderBy=name` `ASC`.

HTML headers

Eloqua's bulk API supports most common HTML headers: in addition to the authentication headers for users of *Basic Authentication*, the `Content-Type` and `Accept` headers specify data formats for the data you import into Eloqua and the bulk API's response data.

Content-Type

The bulk API supports both JSON and CSV file formats as data sources for `PUT` and `POST` requests. The bulk API **does not** support XML. For `PUT` and `POST` requests, you **must** specify either `application/json` or `text/csv` in the `Content-Type` header: if you do not include the Content-Type header an error will occur. The Content-Type header is not required for `GET` and `DELETE` requests, because these do not accept data inputs.

Accept

The `Accept` parameter specifies what file formats you, the client, are willing to accept from the server: either JSON or CSV. Use of the accept header is optional: if you do not specify a parameter for `Accept`, the response will default to JSON.

The following call requests the list of Contact fields in CSV format:

```
GET https://<host>.eloqua.com/api/bulk/2.0/contacts/fields
Accept: text/csv
```

The response would resemble:

```
name,internalName,dataType,defaultValue,hasReadOnlyConstraint,hasNotNullConstraint
Email Address,C_EmailAddress,emailAddress,,False,False
First Name,C_FirstName,string,,False,False
```

The following requests the list of contact fields in JSON format:

```
GET https://<host>.eloqua.com/api/bulk/2.0/contacts/fields
Accept: application/json
```

The response resembles:

```
{
   "count": 82,
   "hasMore": false,
   "items": [
      {
         "createdAt": "1900-01-01T05:00:00.0000000Z",
         "dataType": "emailAddress",
         "hasNotNullConstraint": false,
         "hasReadOnlyConstraint": false,
         "hasUniquenessConstraint": true,
         "internalName": "C_EmailAddress",
         "name": "Email Address",
         "statement": "{{Contact.Field(C_EmailAddress)}}",
         "updatedAt": "1900-01-01T05:00:00.0000000Z",
         "uri": "/contacts/fields/100001"
      },
      {
         "createdAt": "1900-01-01T05:00:00.0000000Z",
         "dataType": "string",
         "hasNotNullConstraint": false,
         "hasReadOnlyConstraint": false,
         "hasUniquenessConstraint": false,
         "internalName": "C_FirstName",
         "name": "First Name",
         "statement": "{{Contact.Field(C_FirstName)}}",
         "updatedAt": "2013-03-26T21:32:13.2530000Z",
         "updatedBy": "Docs.Example",
         "uri": "/contacts/fields/100002"
      }
   ]
}
```


Data Encoding

When uploading data, Eloqua expects data to be encoded with UTF-8.

Example call and response

You can create bulk API requests using whatever language you wish. The examples and tutorials in this guide show the basic request that you need to send to perform

your request, ignoring language-specific code examples.

Request:

```
https://<host>.eloqua.com/API/Bulk/2.0/contact/import/123
```

Response:

```
{
   "createdAt": "2014-05-13T18:25:32.0270000Z",
   "createdBy": "Docs.Example",
   "fields": {
      "C_EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
      "C_FirstName": "{{Contact.Field(C_FirstName)}}",
      "C_LastName": "{{Contact.Field(C_LastName)}}"
   },
   "identifierFieldName": "C_EmailAddress",
   "isSyncTriggeredOnImport": false,
   "isUpdatingMultipleMatchedRecords": false,
   "name": "Docs Import Example",
   "syncActions": [],
   "updatedAt": "2014-05-13T18:25:32.0270000Z",
   "updatedBy": "Docs.Example",
   "uri": "/contacts/imports/1183"
}
```

Eloqua elements

The bulk API enables you to import and export data for numerous elements within Eloqua. The bulk API also has its own elements that it uses to interact with Eloqua.

The following sections describe all of the elements encountered when using the bulk API, and highlights the different interaction patterns associated with each type.

Each Eloqua element type has its own set of associated fields stored in Eloqua.

The bulk API exists to allow you to import and export data related to these elements. However, due to the asynchronous design of the bulk API, you never interact directly with the Eloqua elements. Rather, you move data into a *staging area* and then synchronize the data into Eloqua or out for your use.

The Eloqua elements are:

- Contacts
- Accounts
- Custom objects
- Activity
- External activities
- Events
- Campaign responses
- Opportunities

With the exception of activity and external activities, you can use a `/fields` endpoint to retrieve the list of fields associated with each element in your Eloqua instance. See Field names for more information.

Activity behaves differently than the other Eloqua elements. Each **Activity Type** has its own set of fields. Refer to Activity Fields for a full description of the activity types that the bulk API supports and their specific fields.

Bulk API elements

The bulk API elements are objects that you create in order to interact with Eloqua through the bulk API. The are, essentially, metadata elements: elements that provide

context to Eloqua and to the bulk API to allow them to perform the appropriate actions.

These bulk elements are:

- Imports
- Exports
- Syncs

Unlike the Eloqua elements, which you interact with indirectly, the bulk API enables you to perform direct CRUD operations on the metadata. If you `GET` an import or an export, Eloqua returns its data. In contrast, you cannot directly `GET` contact or activity data: you have to go through the export steps to obtain it.

Export data from Eloqua

The process for exporting data from Eloqua using the bulk API follows 3 basic steps:

1. Create the export definition
2. Synchronize the outgoing data into a temporary staging area
3. Retrieve the data

Eloqua supports export definitions for Activities, Accounts, Contacts, Custom objects, Events, and Campaign responses.

In this tutorial:

Important: If you are working with large volumes of data, you should break your dataset into smaller segments using filters. Learn more about Bulk API limits.

To export data from Eloqua

1. Create the export definition, which outlines your request's details. Once you create it, Eloqua will respond with a URI, which you will need in order to synchronize the data for export (step 2).

   Definition parameters

Name	Required/Optional	Description
`name`	Required	The name of your export for reference
`fields`	Required	This is the most important parameter in your export definition. It defines which fields of the Eloqua record you are exporting will be mapped to the

Name	Required/Optional	Description
		export output. A field parameter is made up of a name value and a field statement, for example: `"ActivityId":"{{Activity.Id}}"`. The name value can be anything you want and doesn't need to match the Eloqua field name. The statement on the other hand needs to be valid Eloqua markup language and must match an existing field for the record being exported. You can retrieve detailed field information by calling the *fields* endpoint. Learn more about retrieving fields information.

Name	Required/Optional	Description
`filter`	Optional	Filters allow you to select the exact data that you wish to export. For instance if you only wish to export contact records where the company field is of a certain value (such as Oracle employees) you can do so using a filter. Learn more about bulk API filtering.
`syncActions`	Optional	Sync Actions are parameters that you declare in an import or export definition that specify additional actions Eloqua should take when you import or export data. Learn more about sync actions.
`areSystemTimestampsInUTC`	Optional	Enables you to

Name	Required/Optional	Description
		control whether system timestamps will be exported in coordinated universal time (UTC). Learn more.

The following export definition requests all Activity records where the activity type is `FormSubmit`. There are different endpoints depending on whether you're creating an export definition for Activities, Accounts, Contacts, Custom objects, Events, or Campaign responses.

Request:

```
POST /bulk/2.0/activities/exports
```

Request body:

```
{
  "name":"Example Activity Export",
  "fields":{
    "ActivityId":"{{Activity.Id}}",
    "AssetName":"{{Activity.Asset.Name}}",
    "ActivityType":"{{Activity.Type}}",
    "ActivityDate":"{{Activity.CreatedAt}}",
    "ContactId":"{{Activity.Contact.Id}}",
    "VisitorId":"{{Activity.Visitor.Id}}",
    "AssetType":"{{Activity.Asset.Type}}",
```

```
    "AssetId":"{{Activity.Asset.Id}}",
    "RawData":"{{Activity.Field(RawData)}}"
  },
  "filter":"'{{Activity.Type}}'='FormSubmit'"
}
```

A successful response will include the export definition's `name`, `fields` and `filter` parameters, along with the new `uri` field, the name of the user who created and updated it, and timestamps for the creation and update.

Response:

```
{
  "name":"Example Activity Export",
  "fields":{
    "ActivityId":"{{Activity.Id}}",
    "AssetName":"{{Activity.Asset.Name}}",
    "ActivityType":"{{Activity.Type}}",
    "ActivityDate":"{{Activity.CreatedAt}}",
    "ContactId":"{{Activity.Contact.Id}}",
    "VisitorId":"{{Activity.Visitor.Id}}",
    "AssetType":"{{Activity.Asset.Type}}",
    "AssetId":"{{Activity.Asset.Id}}",
    "RawData":"{{Activity.Field(RawData)}}"
  },
  "filter":"'{{Activity.Type}}'='FormSubmit'",
  "dataRetentionDuration":"P7D",
  "uri":"/activities/exports/26331",
```

```
  "createdBy":"API.User",
  "createdAt":"2015-07-16T19:14:13.5598178Z",
  "updatedBy":"API.User",
  "updatedAt":"2015-07-16T19:14:13.5598178Z"
}
```

Save the `uri` field value for the next step, in this case: `/activities/exports/26331`.

2. Synchronize the data. Once you have your export definition's `uri`, you can use it to synchronize the requested data. This is done by sending the export definition's uri to the `/syncs` endpoint. Continuing the above example, the request would be:

```
POST /bulk/2.0/syncs
```

Request body:

```
{
  "syncedInstanceUri" : "/activities/exports/26331"
}
```

The response will include a `status` parameter, creation metadata, and the sync `uri`:

```
{
  "syncedInstanceUri":"/activities/exports/26331",
  "status":"pending",
  "createdAt":"2015-07-16T19:25:14.0808068Z",
  "createdBy":"API.User",
  "uri":"/syncs/52524"
}
```

The sync `status` is `pending`. As the sync progresses, the `status` will change. You'll need to check the status of the sync by sending a GET request to the sync's `uri` using the Retrieve a sync endpoint.

> **Tip**: To avoid polling to check the sync's status, the `syncs` endpoint has an optional `callbackUrl` parameter available. When a sync completes, Eloqua will make a call to the URL specified in this parameter. Learn more about the sync endpoint.

Send a GET request to retrieve the sync using the sync's `uri` . Continuing the example above:

```
GET /bulk/2.0/syncs/52524
```

When the sync is complete there will be a `200 OK` response that resembles the following:

```
{
   "syncedInstanceUri":"/activities/exports/26331",
   "syncStartedAt":"2015-07-16T19:25:14.3900000Z",
   "syncEndedAt":"2015-07-16T19:25:19.9370000Z",
   "status":"success",
   "createdAt":"2015-07-16T19:25:13.6930000Z",
   "createdBy":"API.User",
   "uri":"/syncs/52524"
}
```

3. Retrieve the data you have synchronized using the sync URI's data endpoint.

   Continuing the example, the request would be:

```
GET /bulk/2.0/syncs/52524/data
```

The response will include the requested records within the `items` property:

```
{
   "Count": 1,
   "hasMore": false,
   "limit": 1000,
   "offset": 1,
   "totalResults": 1,
   "items": [
      {
         "ActivityId": "999",
         "AssetName": "Forms-BulkActivity",
         "ActivityType": "FormSubmit",
         "ActivityDate": "2014-01-29 13:45:31",
         "EmailAddress": "allison@example.com",
         "ContactId": "123",
         "VisitorId": "1",
         "AssetType": "Form",
         "AssetId": "1",
      }
   ]
}
```

Note: If you do not specify a `limit` when you send your `GET` request, the `limit` defaults to 1000. You can specify any limit up to 50000, though requesting larger

volumes may create performance issues.

If there were more than 1000 matching items, `hasMore` would be `true` and you would be able to request the next batch of items using an appropriate `offset`. For example, if there were 1500 items, the following call would request items 1000 through 1500:

```
GET /bulk/2.0/syncs/52524/data?offset=1000
```

For more information on how to use these query parameters, see Retrieving large volumes of data.

See the syncs data endpoint documentation for full list of URL parameters.

Data Export Timeouts

- Contacts, Activities, Custom Objects, and Events: 6 hours
- Accounts and Campaign Responses: 1 hour

Import data into Eloqua

To import data into Eloqua using the bulk API:

1. Create the import definition.
2. Upload the incoming data into a temporary staging area.
3. Synchronize the data from the staging area into Eloqua.

Eloqua supports import definitions for activities, accounts, contacts, custom objects, and events. Email addresses are not included in this list of import definitions because they can only be used with sync actions.

Important: If you are working with large volumes of data, you should break your dataset into smaller segments using filters. Learn more about Bulk API limits.

To import data into Eloqua

1. Create the import definition.

   The import definition outlines your request's details. Once you create an import definition, Eloqua will respond with a `uri` which you will need in order to upload data into Eloqua (step 2).

   Definition parameters

Name	Required/Optional	Description
`name`	Required	The name of your import for reference
`fields`	Required	This is the most important parameter in your import definition. It defines which fields your imported data will be mapped to for the given Eloqua record type. A field parameter is made up of a name value and a field statement, for example: `"ActivityId":"`

Name	Required/Optional	Description
		`{{Activity.Id}}"`. The name value can be anything you want and doesn't need to match the Eloqua field name. The statement on the other hand needs to be valid Eloqua markup language and must match an existing field for the record being exported. You can retrieve detailed field information by calling the *fields* endpoint. Learn more about retrieving fields information. **Note**: Lead scoring fields are read only and can only be used in Contact Exports.
`identifierFieldName`	Required	This parameter is used to check the imported data against existing records within Eloqua. It should match one of the field names as defined in the above `fields` parameter.

Name	Required/Optional	Description
		Choose a field that is likely to be unique to avoid updating the wrong record. Do not use a large text field as the `identifierFieldName` as this can produce errors.
`isSyncTriggeredOnImport`	Optional	A Boolean parameter specifying whether the sync is triggered automatically upon import. If set to `true`, the Bulk API automatically syncs your data into Eloqua when you upload it to the staging area. If set to `false`, you must manually create the sync operation that merges your data into Eloqua. Manually syncing the data provides more control over the timing of the syncs, and allows you to break large sync operations into smaller batches. This can mitigate performance issues in your Eloqua instance. By default, `isSyncTriggeredOnImport` is set to `false`.

Name	Required/Optional	Description
`dataRetentionDuration`	Optional	The length of time that unsync'd data from this import should remain in the staging area. Bulk API 2.0 uses the ISO-8601 standard for specifying all durations. Valid values are anything from PT1H (1 hour) to P14D (2 weeks). This setting will default to P7D (7 days) if not explicitly set during definition creation.
`syncActions`	Optional	Sync Actions are parameters that you declare in an import or export definition that specify additional actions Eloqua should take when you import or export data. Learn more about sync actions.
`isUpdatingMultipleMatchedRecords`	Optional	Whether or not imported data will be mapped to multiple existing records. Learn more.
`importPriorityUri`	Optional	Must reference an existing /imports/priorities/{id}.
`linkUsers`	Optional	Enables you to link users from a CRM to your Eloqua accounts during an account import.

Name	Required/Optional	Description
		Learn more.
`updateRuleByField`	Optional	Specifies how Eloqua should handle updating records when you import a new value for an existing field by field. Learn more.

The following import definition will map incoming data to Eloqua contact record fields, specifically `firstName`, `lastName`, and `emailAddress`. There are different endpoints depending on whether you're creating an import definition for Activities, Accounts, Contacts, Custom objects, or Events.

> **Note**: Creating an import definition is required if you are updating `fields` for the Eloqua record type. If you are not updating `fields`, the sync action definition endpoints can be used to specify up to 10 actions Eloqua should take without importing data. Using the sync action definition endpoints is always recommended if no updates are required, because it is a quicker operation than importing data. There are different endpoints for accounts, contacts, and custom objects. For a full list of available actions, see Sync actions.

Request:

```
POST /bulk/2.0/contacts/imports/
```

Request body:

```
{
  "name": "Docs Import Example",
  "fields": {
     "firstName": "{{Contact.Field(C_FirstName)}}",
     "lastName": "{{Contact.Field(C_LastName)}}",
     "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "identifierFieldName": "emailAddress",
   "isSyncTriggeredOnImport" : "false"
}
```

A successful response will include the import definition's `name`, `fields`, `identifierFieldName`, `isSyncTriggeredOnImport`, `isUpdatingMultipleMatchedRecords`, along with the new `uri` field, the name of the user who created and updated it, and timestamps for the creation and update.

Response:

```
{
    "name": "Docs Import Example",
    "fields": {
        "firstName": "{{Contact.Field(C_FirstName)}}",
        "lastName": "{{Contact.Field(C_LastName)}}",
        "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
    },
    "identifierFieldName": "emailAddress",
    "isSyncTriggeredOnImport": false,
    "isUpdatingMultipleMatchedRecords": false,
    "uri": "/contacts/imports/1182",
```

```
  "createdBy": "Docs.Example",
  "createdAt": "2014-05-13T14:13:30.0402961Z",
  "updatedBy": "Docs.Example",
  "updatedAt": "2014-05-13T14:13:30.0402961Z"
}
```

You will use the returned `uri` parameter to upload data into Eloqua.

2. Upload the data: Once you have your import definition's `uri`, you can use it to upload the incoming data.

> **Note**: When uploading data, Eloqua expects data to be encoded with UTF-8.

Continuing the above example, the request would be:

```
POST /bulk/2.0/contacts/imports/1182/data
```

Request body:

```
[
  {
    "firstName": "Juan",
    "lastName": "Garcia",
    "emailAddress": "juan@example.com"
  },
  {
    "firstName": "Tatiana",
```

```
    "lastName": "Smirnov",
    "emailAddress": "tatiana@example.com"
  }
]
```

The response, if successful is an `HTTP 204 No Content` message. The data is now in the staging area, ready to be synced into Eloqua.

> **Note**: There is a hard limit of 32 MB per request when posting data to an import sync's staging area. Requests larger than 32 MB will fail by returning a 404 Not Found response. To import more than 32 MB of data, you can make multiple posts of 32 MB (or less) to the import definition before syncing. It is recommended to sync an import whenever approximately 250 MB of data has been posted to its staging area.

3. Synchronize the data: You can synchronize the data from the staging area into your Eloqua database.

> **Note**: If `isSyncTriggeredOnImport` was set to `true`, the sync occurs automatically and you can skip this step. If `isSyncTriggeredOnImport` is set to `false`, you must manually create the sync. If uploading more than 32 MB of data within a sync, `isSyncTriggeredOnImport` needs to be set to `false` to allow multiple requests for posting data.

Continuing the above example, the request would be:

```
POST /bulk/2.0/syncs
```

Request body:

```
{
  "syncedInstanceUri": "/contacts/imports/1182"
}
```

The response will include a `status` parameter, creation metadata, and the sync `uri`:

```
{
  "syncedInstanceUri": "/contacts/imports/1182",
  "status": "pending",
  "createdAt": "2014-05-13T17:58:34.0176959Z",
  "createdBy": "Docs.Example",
  "uri": "/syncs/1208"
}
```

The sync `status` is `pending`. As the sync progresses, the `status` message will change.

Checking the status of the sync

To check the status of the sync, send a GET request to the sync's `uri`. In this example:

```
GET /bulk/2.0/syncs/1208
```

When the sync is complete there will be a `200 OK` response that resembles the following:

```
{
  "syncedInstanceUri": "/contacts/imports/1182",
  "syncStartedAt": "2014-05-13T17:58:34.2770000Z",
```

```
  "status": "success",
  "createdAt": "2014-05-13T17:58:33.8130000Z",
  "createdBy": "Docs.Example",
  "uri": "/syncs/1208"
}
```

> **Tip**: To avoid polling to check the sync's status, the `syncs` endpoint has an optional `callbackUrl` parameter available. When a sync completes, Eloqua will make a call to the URL specified in this parameter. Learn more about the sync endpoint.

Importing account data

This tutorial will go into detail on how to import **account data** into Eloqua, and provide example use cases for doing so. The Import data into Eloqua tutorial provides a general example of how to import all data types such as activity data, account data, contact data, custom object data, and event data into Eloqua. Refer to Import data into Eloqua for general rules for importing data.

Use case: Importing accounts and linking users to accounts

The following example demonstrates how to import accounts and link users to accounts.

To import accounts and link users to accounts:

1. Create the account import definition.

   Request:

```
POST /bulk/2.0/accounts/imports
```

Request body:

In the request body we will set `linkUsers` to `true`, to indicate we will be linking users to accounts.

```
{
  "name": "Import Accounts and link to Users",
  "fields": {
    "CompanyName": "{{Account.Field(M_CompanyName)}}",
    "UserCRMIdLink": "{Account.User.Field(CRMOSCUserId)}}"
  },
  "identifierFieldName": "CompanyName"
  "linkUsers": "true"
}
```

A successful response will include the import definition's `name`, `fields`, `identifierFieldName`, `isSyncTriggeredOnImport`, `isUpdatingMultipleMatchedRecords`, `linkUsers`, along with the new `uri` field, the name of the user who created and updated it, and timestamps for the creation and update.

Response:

```
{
  "linkUsers": true,
  "name": "Import Accounts and link to Users",
  "fields": {
    "CompanyName": "{{Account.Field(M_CompanyName)}}",
```

```
    "UserCRMIdLink": "{{Account.User.Field(CRMOSCUserId)}}"
  },
  "identifierFieldName": "CompanyName",
  "isSyncTriggeredOnImport": false,
  "dataRetentionDuration": "P7D",
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/accounts/imports/43",
  "createdBy": "API.User",
  "createdAt": "2020-11-09T17:35:22.6488629Z",
  "updatedBy": "API.User",
  "updatedAt": "2020-11-09T17:35:22.6488629Z"
}
```

You will use the returned `uri` parameter to upload data into Eloqua.

2. Upload the account data into a temporary staging area, using the `uri` from step 1.

> **Note**: When uploading data, Eloqua expects data to be encoded with UTF-8.

Request:

```
POST /bulk/2.0/accounts/imports/43/data
```

Request body:

In the request body we can specify multiple User IDs to upload to the account, separated by commas. Note this applies only when using the **UserCRMIdLink** field.

```
[
 {
  "CompanyName": "Eloqua",
  "UserCRMIdLink": "754a1a4baadb4ace991ee,754a1a4baadb4ace991ea"
 }
]
```

Response:

```
204 No Content
```

The response, if successful is an `HTTP 204 No Content` message. The data is now in the staging area, ready to be synced into Eloqua.

3. Synchronize the data. You can synchronize the data from the staging area into your Eloqua database.

   Continuing the above example, the request would be:

   ```
   POST /bulk/2.0/syncs
   ```

   Request body:

   ```
   {
    "syncedInstanceUri": "/accounts/imports/43"
   }
   ```

   The response will include a `status` parameter, creation metadata, and the sync `uri`:

```
{
  "syncedInstanceUri": "/accounts/imports/43",
  "status": "pending",
  "createdAt": "2020-11-09T19:16:50.2642759Z",
  "createdBy": "API.User",
  "uri": "/syncs/24"
}
```

The sync `status` is `pending`. As the sync progresses, the `status` message will change.

Checking the status of the sync

To check the status of the sync, send a GET request to the sync's `uri`. In this example:

```
GET /bulk/2.0/syncs/24
```

When the sync is complete there will be a `200OK` response that resembles the following:

```
{
  "syncedInstanceUri": "/accounts/imports/43",
  "syncStartedAt": "2020-11-09T19:16:50.6700000Z",
  "syncEndedAt": "2020-11-09T19:16:52.5370000Z",
  "status": "success",
  "createdAt": "2020-11-09T19:16:50.2830000Z",
  "createdBy": "API.User",
  "uri": "/syncs/24"
}
```

> **Tip**: To avoid polling to check the sync's status, the `syncs` endpoint has an optional `callbackUrl` parameter available. When a sync completes, Eloqua will make a call to the URL specified in this parameter. Learn more about the sync endpoint.

Using import and export parameters

Import and export parameters enable you to control and describe your imports and exports with the Bulk API. This tutorial will walkthrough how to use the following parameters:

- `mapDataCards` (Imports)
- `syncActions`(Imports and Exports)
- `importPriorityUri` (Imports)
- `updateRule` (Imports)
- `areSystemTimestampsInUTC` (Exports)
- `linkUsers` (Imports)
- `updateRuleByField` (Imports)

All of these parameters are optional. See Import Characteristics or Export Characteristics for a full list of available parameters.

mapDataCards

The `mapDataCards` parameter enables you to decide whether or not custom object records or event registrants will be mapped on import. If set to `true`, you'll need to specify the Eloqua entity field for mapping and the field that will be used for matching.

Here are the parameters you'll need to familiarize yourself with:

- `mapDataCards` (**boolean**): Whether or not custom object records or event registrants will be mapped on import. By default, `mapDataCards` is `false`. If you set it to `true`, you must specify the fields for mapping.
- `mapDataCardsEntityField` (**string**): Specifies which Eloqua entity field will be used for mapping.
- `mapDataCardsSourceField` (**string**): Specifies the source field that will be used for matching.
- `mapDataCardsEntityType` (**string**): Specifies the entity of the custom object record or event import. Allowed values are "Contact" or "Company".
- `mapDataCardsCaseSensitiveMatch` (**boolean**): Whether to perform a case sensitive search when mapping custom object records or events to a contact or account.

Example

Let's create a new custom object import and map records to contacts using the contact field `ContactIdExt`:

HTTP Request

```
POST /api/bulk/2.0/customObjects/8/imports
Content-Type: application/json
```

Request Body

```
{
  "name": "Simple CDO Contact Import 2",
  "mapDataCards": "true",
  "mapDataCardsEntityField": "{{Contact.Field(ContactIDExt)}}",
  "mapDataCardsSourceField": "ContactIDExt",
  "mapDataCardsEntityType": "Contact",
  "mapDataCardsCaseSensitiveMatch": "false",
  "updateRule": "always",
  "dataRetentionDuration": "P7D",
```

```
  "fields": {
    "email": "{{CustomObject[8].Field[58]}}",
    "first": "{{CustomObject[8].Field[405]}}",
    "contactIDExt": "{{CustomObject[8].Field[406]}}"
  },
  "identifierFieldName": "contactIDExt"
}
```

- `"mapDataCardsEntityField": "{{Contact.Field(ContactIDExt)}}"`

  Because we want to map records to contacts using their contact ID, we are setting `mapDataCardsEntityField` to `{{Contact.Field(ContactIDExt)}}` because `{{Contact.Field(ContactIDExt)}}` is the internal Eloqua name for the field.

- `"mapDataCardsSourceField": "ContactIDExt"`

  The source field from the imported data that we will use for mapping is `ContactIDExt`, so we have set `mapDataCardsSourceField` to `ContactIDExt`.

- `"contactIDExt": "{{CustomObject[8].Field[406]}}"`

  Whichever field you use for mapping has to exist in both entities (in this case on a contact record and CDO record). Meaning we have to create a new custom object data field for the `contactIDExt` value.

Response

```
{
  "id": 836,
  "parentId": 8,
  "mapDataCards": true,
  "mapDataCardsCaseSensitiveMatch": false,
  "mapDataCardsEntityField": "{{Contact.Field(ContactIDExt)}}",
  "mapDataCardsSourceField": "ContactIDExt",
```

```
  "mapDataCardsEntityType": "Contact",
  "name": "Simple CDO Contact Import 2",
  "updateRule": "always",
  "fields": {
    "email": "{{CustomObject[8].Field[58]}}",
    "first": "{{CustomObject[8].Field[405]}}",
    "ContactIDExt": "{{CustomObject[8].Field[406]}}"
  },
  "identifierFieldName": "ContactIDExt",
  "isSyncTriggeredOnImport": false,
  "dataRetentionDuration": "P7D",
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/customObjects/8/imports/836",
  "createdBy": "API.User",
  "createdAt": "2016-02-01T21:43:03.7931253Z",
  "updatedBy": "API.User",
  "updatedAt": "2016-02-01T21:43:03.7931253Z"
}
```

Example

You may also want to include contact fields in the custom object import. We will use the `mapDataCardsEntityField` parameter and match on contact email address. When updating contacts, you should always match on email address, as Eloqua will try to create new contacts if the email address doesn't already exist.

Note: When using the Bulk API, entities related to core entities are available as long as the relationship resolves to a single entity and not many. For example, Account fields could be included in Contact exports and Contact fields could be included in Custom Object exports, as the relationship in these scenarios always resolves to a single entity. Custom Objects do not always resolve to a single

entity, i.e. there could be more than one Custom Object record mapped to a Contact, so Custom Object fields cannot be included in Contact exports. Read more within the Eloqua Markup Language version 2 reference page under the **Available entities** section.

Let's create a new custom object import that includes contact fields:

HTTP Request

```
POST /api/bulk/2.0/customObjects/8/imports
Content-Type: application/json
```

Request Body

```
{
 "mapDataCards": true,
 "mapDataCardsCaseSensitiveMatch": false,
 "mapDataCardsEntityField": "{{Contact.Field(C_EmailAddress)}}",
 "mapDataCardsSourceField": "customObjectEmail",
 "mapDataCardsEntityType": "Contact",
 "name": "Custom Object Import with Contact Updates",
 "updateRule": "always",
 "fields": {
  "customObjectEmail": "{{CustomObject[8].Field[103]}}",
  "contactFirstName": "{{CustomObject[8].Contact.Field(C_FirstName)}}"
 },
 "identifierFieldName": "customObjectEmail",
 "isSyncTriggeredOnImport": true
}
```

syncActions

The `syncActions` parameter enables you to declare in an import or export definition that Eloqua should perform additional actions when you importing or exporting data.

Using a sync action parameter generally involves specifying the `action` to perform and the `destination` for the entities.

See Sync actions parameters to see a full list of actions you can perform using this parameter.

Example

Create a contact import definition and use a sync action to add contacts to a contact list:

HTTP Request

```
POST /api/bulk/2.0/contacts/imports
Content-Type: application/json
```

Request Body

```
{
  "name": "Email Address Import",
  "fields": {
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
  },
  "identifierFieldName": "EmailAddress",
  "syncActions": [
    {
      "destination": "{{ContactList[31]}}",
      "action": "add"
    }
  ]
}
```

Response

```
{
   "name": "Email Address Import",
   "fields": {
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "identifierFieldName": "EmailAddress",
   "syncActions": [
      {
         "destination": "{{ContactList[31]}}",
         "action": "add"
      }
   ],
   "isSyncTriggeredOnImport": false,
   "dataRetentionDuration": "P7D",
   "isUpdatingMultipleMatchedRecords": false,
   "uri": "/contacts/imports/3",
   "createdBy": "API.User",
   "createdAt": "2018-01-29T14:29:15.2227712Z",
   "updatedBy": "API.User",
   "updatedAt": "2018-01-29T14:29:15.2227712Z"
}
```

importPriorityUri

The `importPriorityUri` parameter is used to set the import to a different priority which overrides the default bulk API priority set in your Eloqua instance. You're able to discover your import priorities via this endpoint,

If you are not familiar with how Eloqua handles data import priorities, see Data import priority for more information.

Example

Create a new import and set the data import priority to #1.

HTTP request

```
POST /api/bulk/2.0/contacts/imports
Content-Type: application/json
```

Request body

```
{
  "name":"Email Address Import",
  "importPriorityUri":"/imports/priorities/100001",
  "fields":{
    "EmailAddress":"{{Contact.Field(C_EmailAddress)}}"
  },
 "identifierFieldName": "EmailAddress"
}
```

Response

```
{
 "name": "Email Address Import",
 "importPriorityUri": "/imports/priorities/100001",
 "fields": {
  "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "identifierFieldName": "EmailAddress",
 "isSyncTriggeredOnImport": false,
 "dataRetentionDuration": "P7D",
 "isUpdatingMultipleMatchedRecords": false,
 "uri": "/contacts/imports/15",
 "createdBy": "API.User",
 "createdAt": "2018-01-24T15:29:02.5037857Z",
 "updatedBy": "API.User",
 "updatedAt": "2018-01-24T15:29:02.5037857Z"
}
```

updateRule

Import definitions include an `updateRule` parameter, which specifies how Eloqua should handle updating records when you import a new value for an existing field. For example, if contact Sally Jones's email address in Eloqua is

`sally.jones9@example.com`, and you import a new email address for Sally Jones, the `updateRule` determines whether Eloqua should retain the existing email address or replace it with the new one. If you do not specify an `updateRule`, the rule type defaults to `always`.

The following rule types are available:

- `always`: Always update.
- `ifNewIsNotNull`: Update if the new value is not blank.
- `ifExistingIsNull`: Update if the existing value is blank.
- `useFieldRule`: Use the rule defined at the field level.

Example

Create a new contact import definition and set the update rule to always replace existing records with new values.

HTTP request

```
POST /api/bulk/2.0/contacts/imports
Content-Type: application/json
```

Request body

```
{
  "name": "Email Address Import",
  "fields": {
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "FirstName": "{{Contact.Field(C_FirstName)}}",
    "LastName": "{{Contact.Field(C_LastName)}}",
    "Company": "{{Contact.Field(C_Company)}}",
    "Address1": "{{Contact.Field(C_Address1)}}",
    "Address2": "{{Contact.Field(C_Address2)}}",
```

```
    "City": "{{Contact.Field(C_City)}}",
    "StateProvince": "{{Contact.Field(C_State_Prov)}}"
  },
  "identifierFieldName": "EmailAddress",
  "updateRule": "always"
}
```

Response

```
{
  "name": "Email Address Import",
  "updateRule": "always",
  "fields": {
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "FirstName": "{{Contact.Field(C_FirstName)}}",
    "LastName": "{{Contact.Field(C_LastName)}}",
    "Company": "{{Contact.Field(C_Company)}}",
    "Address1": "{{Contact.Field(C_Address1)}}",
    "Address2": "{{Contact.Field(C_Address2)}}",
    "City": "{{Contact.Field(C_City)}}",
    "StateProvince": "{{Contact.Field(C_State_Prov)}}"
  },
  "identifierFieldName": "EmailAddress",
  "isSyncTriggeredOnImport": false,
  "dataRetentionDuration": "P7D",
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/contacts/imports/6",
  "createdBy": "API.User",
  "createdAt": "2018-01-29T14:44:42.7717370Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-01-29T14:44:42.7717370Z"
}
```

areSystemTimestampsInUTC

The boolean parameter `areSystemTimestampsInUTC` enables you to control whether system timestamps will be exported in coordinated universal time (UTC).

Example

With `areSystemTimestampsInUTC` set to `true`, export accounts that have been created after September 9, 2016 13:46:20.975 UTC:

HTTP request

```
POST /api/bulk/2.0/accounts/exports
Content-Type: application/json
```

Request body

```
{
  "name": "System Timestamps in UTC Export",
  "fields": {
    "companyName": "{{Account.Field(M_CompanyName)}}",
    "createdDate": "{{Account.Field(M_DateCreated)}}",
    "modifiedDate": "{{Account.Field(M_DateModified)}}"
  },
  "areSystemTimestampsInUTC": true,
  "filter": "'{{Account.Field(M_DateCreated)}}' > '2016-09-01 13:46:20.975'"
}
```

Response

```
{
  "name": "System Timestamps in UTC Export",
  "fields": {
    "companyName": "{{Account.Field(M_CompanyName)}}",
    "createdDate": "{{Account.Field(M_DateCreated)}}",
    "modifiedDate": "{{Account.Field(M_DateModified)}}"
  },
  "filter": "'{{Account.Field(M_DateCreated)}}' > '2016-09-01 13:46:20.975'",
  "dataRetentionDuration": "PT12H",
  "areSystemTimestampsInUTC": true,
  "uri": "/accounts/exports/92",
  "createdBy": "API.User",
```

```
  "createdAt": "2020-10-30T00:31:10.4176409Z",
  "updatedBy": "API.User",
  "updatedAt": "2020-10-30T00:31:10.4176409Z"
}
```

linkUsers

The `linkUsers` parameter is a boolean parameter that enables you to link users from a CRM to your Eloqua accounts during an account import.

If `linkUsers` is set to `true` one of the following CRM User ID fields is required in the Account import definition:

```
CRMOSCUserId
CRMSfdcUserId
CRMMSDUserId
CRMSODUserId
```

The statement formatting is:

`{{Account.User.Field(<Field Name>)}}`

Where `<Field Name>` is the CRM User ID field.

Example

Create an Account Import definition, and link users using the field "Oracle Sales Cloud User ID" (CRMOSCUserId):

HTTP request

```
POST /api/bulk/2.0/accounts/imports
Content-Type: application/json
```

Request body

```
{
  "name": "Import Accounts and link to Users",
  "fields": {
    "CompanyName": "{{Account.Field(M_CompanyName)}}",
    "UserCRMIdLink": "{{Account.User.Field(CRMOSCUserId)}}"
  },
  "identifierFieldName": "CompanyName"
  "linkUsers": "true"
}
```

Response

```
{
  "linkUsers": true,
  "name": "Import Accounts and link to Users",
  "fields": {
    "CompanyName": "{{Account.Field(M_CompanyName)}}",
    "UserCRMIdLink": "{{Account.User.Field(CRMOSCUserId)}}"
  },
  "identifierFieldName": "CompanyName",
  "isSyncTriggeredOnImport": false,
  "dataRetentionDuration": "P7D",
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/accounts/imports/41",
  "createdBy": "API.User",
  "createdAt": "2020-10-29T18:33:18.8585154Z",
  "updatedBy": "API.User",
  "updatedAt": "2020-10-29T18:33:18.8585154Z"
}
```

updateRuleByField

Import definitions include an `updateRuleByField` parameter, which specifies how Eloqua should handle updating records when you import a new value for an existing field by field. If you do not specify an `updateRuleByField`, or leave a field out of `updateRuleByField`, the rule type defaults to the rule set with updateRule. The following rule types are available:

- `always`: Always update.
- `ifNewIsNotNull`: Update if the new value is not blank.
- `ifExistingIsNull`: Update if the existing value is blank.
- `useFieldRule`: Use the rule defined at the field level.

Example

Create a new contact import definition and set the update rule by field, for some fields.

HTTP request

```
POST /api/bulk/2.0/accounts/imports
Content-Type: application/json
```

Request body

```
{
  "name": "updateRuleByField Import",
  "fields": {
    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "FirstName": "{{Contact.Field(C_FirstName)}}",
    "LastName": "{{Contact.Field(C_LastName)}}",
    "Company": "{{Contact.Field(C_Company)}}",
    "Address1": "{{Contact.Field(C_Address1)}}",
    "Address2": "{{Contact.Field(C_Address2)}}",
    "City": "{{Contact.Field(C_City)}}",
    "StateProvince": "{{Contact.Field(C_State_Prov)}}"
  },
  "identifierFieldName": "EmailAddress",
  "updateRule": "always",
  "updateRuleByField": {
    "FirstName": "ifNewIsNotNull",
    "LastName": "useFieldRule",
    "Company": "ifExistingIsNull"
  }
```

```
}
```

Response

```
{
   "name": "updateRuleByField Import",
   "updateRule": "always",
   "fields": {
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
      "FirstName": "{{Contact.Field(C_FirstName)}}",
      "LastName": "{{Contact.Field(C_LastName)}}",
      "Company": "{{Contact.Field(C_Company)}}",
      "Address1": "{{Contact.Field(C_Address1)}}",
      "Address2": "{{Contact.Field(C_Address2)}}",
      "City": "{{Contact.Field(C_City)}}",
      "StateProvince": "{{Contact.Field(C_State_Prov)}}"
   },
   "updateRuleByField": {
      "FirstName": "ifNewIsNotNull",
      "LastName": "useFieldRule",
      "Company": "ifExistingIsNull"
   },
   "identifierFieldName": "EmailAddress",
   "isSyncTriggeredOnImport": false,
   "dataRetentionDuration": "P7D",
   "isUpdatingMultipleMatchedRecords": false,
   "uri": "/contacts/imports/854",
   "createdBy": "API.User",
   "createdAt": "2022-02-24T23:48:28.2330000Z",
   "updatedBy": "API.User",
   "updatedAt": "2022-02-24T23:48:28.2330000Z"
}
```

Fields (metadata)

The bulk API provides fields endpoints for account, contact, custom objects, events, campaign responses, and opportunities. Activity elements behave differently from other Eloqua elements. Activity types have their own set of associated fields which are

detailed here. In order to create export or import definitions using the bulk API, you need to include the statement definitions for the fields you want to retrieve or submit (respectively).

Example

Let's say we want to export a set of contacts from Eloqua. Before we can do so, we need to know which fields are available, and what their Eloqua markup language statements are.

Retrieve a list of the first 10 contact field parameters:

```
GET /contacts/fields?limit=10
```

> **Note**: There are likely to be many more fields than 10, and the exact number differs for every Eloqua instance, but for the purposes of this example we will limit our scope to the first 10 fields.

Response:


```
{
   "items":[
      {
         "name":"Email Address",
         "internalName":"C_EmailAddress",
         "dataType":"emailAddress",
         "hasReadOnlyConstraint":false,
         "hasNotNullConstraint":false,
         "hasUniquenessConstraint":true,
         "statement":"{{Contact.Field(C_EmailAddress)}}",
```

```
    "uri":"/contacts/fields/100001",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedAt":"1900-01-01T05:00:00.0000000Z"
  },
  {
    "name":"First Name",
    "internalName":"C_FirstName",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_FirstName)}}",
    "uri":"/contacts/fields/100002",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedBy":"MgrPlatformTeamPod1",
    "updatedAt":"2014-06-16T14:43:50.8100000Z"
  },
  {
    "name":"Last Name",
    "internalName":"C_LastName",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_LastName)}}",
    "uri":"/contacts/fields/100003",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedBy":"MgrPlatformTeamPod1",
    "updatedAt":"2014-06-16T14:43:50.8100000Z"
  },
  {
    "name":"Company",
    "internalName":"C_Company",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_Company)}}",
    "uri":"/contacts/fields/100004",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
```

```
    "updatedBy":"MgrPlatformTeamPod1",
    "updatedAt":"2014-06-16T14:43:50.8030000Z"
  },
  {
    "name":"Email Display Name",
    "internalName":"C_EmailDisplayName",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_EmailDisplayName)}}",
    "uri":"/contacts/fields/100005",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedAt":"1900-01-01T05:00:00.0000000Z"
  },
  {
    "name":"Address 1",
    "internalName":"C_Address1",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_Address1)}}",
    "uri":"/contacts/fields/100006",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedBy":"MgrPlatformTeamPod1",
    "updatedAt":"2014-06-16T14:43:50.7900000Z"
  },
  {
    "name":"Address 2",
    "internalName":"C_Address2",
    "dataType":"string",
    "hasReadOnlyConstraint":false,
    "hasNotNullConstraint":false,
    "hasUniquenessConstraint":false,
    "statement":"{{Contact.Field(C_Address2)}}",
    "uri":"/contacts/fields/100007",
    "createdAt":"1900-01-01T05:00:00.0000000Z",
    "updatedBy":"MgrPlatformTeamPod1",
    "updatedAt":"2014-06-16T14:43:50.7930000Z"
```

```
    },
    {
      "name":"Address 3",
      "internalName":"C_Address3",
      "dataType":"string",
      "hasReadOnlyConstraint":false,
      "hasNotNullConstraint":false,
      "hasUniquenessConstraint":false,
      "statement":"{{Contact.Field(C_Address3)}}",
      "uri":"/contacts/fields/100008",
      "createdAt":"1900-01-01T05:00:00.0000000Z",
      "updatedAt":"1900-01-01T05:00:00.0000000Z"
    },
    {
      "name":"City",
      "internalName":"C_City",
      "dataType":"string",
      "hasReadOnlyConstraint":false,
      "hasNotNullConstraint":false,
      "hasUniquenessConstraint":false,
      "statement":"{{Contact.Field(C_City)}}",
      "uri":"/contacts/fields/100009",
      "createdAt":"1900-01-01T05:00:00.0000000Z",
      "updatedBy":"MgrPlatfoemailrmTeamPod1",
      "updatedAt":"2014-06-16T14:43:50.8000000Z"
    },
    {
      "name":"State or Province",
      "internalName":"C_State_Prov",
      "dataType":"string",
      "hasReadOnlyConstraint":false,
      "hasNotNullConstraint":false,
      "hasUniquenessConstraint":false,
      "statement":"{{Contact.Field(C_State_Prov)}}",
      "uri":"/contacts/fields/100010",
      "createdAt":"1900-01-01T05:00:00.0000000Z",
      "updatedBy":"MgrPlatformTeamPod1",
      "updatedAt":"2014-06-16T14:43:50.8130000Z"
    }
  ],
```

```
    "totalResults":62,
    "limit":10,
    "offset":0,
    "count":10,
    "hasMore":true
}
```

On this list we see `Email Address`, `First Name`, `Last Name`, `Company`, `Email Display Name`, `Address 1`, `Address 2`, `Address 3`, `City` and `State or Province`.

Let's say now we want to export a set of contact containing email address, last name, and city information. To do so, we would create the following export definition:

```
POST /contacts/exports
```

Request body:

```
{
  "name":"New Contact Export",
  "fields":{
    "email_contact":"{{Contact.Field(C_EmailAddress)}}",
    "family_name":"{{Contact.Field(C_LastName)}}",
    "city_they_live_in":"{{Contact.Field(C_City)}}"
  }
}
```

Response:

```
{
    "name":"New Contact Export",
    "fields":{
        "email_contact":"{{Contact.Field(C_EmailAddress)}}",
        "family_name":"{{Contact.Field(C_LastName)}}",
        "city_they_live_in":"{{Contact.Field(C_City)}}"
    },
```

```
    "dataRetentionDuration":"P7D",
    "uri":"/contacts/exports/32855",
    "createdBy":"API.User",
    "createdAt":"2015-09-22T18:57:10.0723218Z",
    "updatedBy":"API.User",
    "updatedAt":"2015-09-22T18:57:10.0723218Z"
}
```

Notes:

- When retrieving the field `M_CompanyName`, the `hasNotNullConstraint` will return `true`, where it should return `false`. The `M_CompanyName` field can be `null`.

Filtering

The bulk API allows you to create filter statements in export definitions. Filters enable you to select the exact data that you wish to export. Filters use the comparison, logic, and existence operators specified by Eloqua expression language to filter export data based on Eloqua field values, existence in contact filters, contact segments, contact lists, account lists, and status in AppCloud services.

Eloqua uses the SQL server implementation of the LIKE operator.

The correct syntax format for Eloqua field values is:

```
'{{EML_field_statement}}' <operator> '<value>'
```

Note that the **field**, expressed in Eloqua markup language, and the **value** are both wrapped in single quotes.

The correct syntax format for using the `EXISTS` existence operator is:

```
EXISTS('{{EML_entity_statement}}')
```

The correct syntax format for using the `STATUS` existence operator is:

```
STATUS('{{EML_service_statement}}') = '<status>'
```

For more detail on the syntax for each entity, see Eloqua expression language.

> **Note**: Filters have a 1000 character limit. Contact fields cannot be used in the filter for activity exports.

Simple filters

Simple filters include one criterion and no logical operators, such as one predicate, filtering on a single field, or one existence operator. The following export definitions illustrate simple filters:

Export contacts whose country is Belgium:

```
{
   "name": "Belgian Contacts Export",
   "fields": {
      "ID": "{{Contact.Id}}",
      "FirstName": "{{Contact.Field(C_FirstName)}}",
      "LastName": "{{Contact.Field(C_LastName)}}",
      "Country": "{{Contact.Field(C_Country)}}"
   },
   "filter": "'{{Contact.Field(C_Country)}}' = 'BE'"
}
```

Export activities of type "`EmailClickthrough`":

```
{
   "name": "Bulk Activity Export - Email Clickthrough",
   "fields": {
      "ActivityId": "{{Activity.Id}}",
      "ActivityType": "{{Activity.Type}}",
      "ActivityDate": "{{Activity.CreatedAt}}",
      "EmailAddress": "{{Activity.Field(EmailAddress)}}",
      "ContactId": "{{Activity.Contact.Id}}",
      "IpAddress": "{{Activity.Field(IpAddress)}}",
      "VisitorId": "{{Activity.Visitor.Id}}",
      "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
      "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
      "AssetType": "{{Activity.Asset.Type}}",
      "AssetName": "{{Activity.Asset.Name}}",
      "AssetId": "{{Activity.Asset.Id}}",
      "SubjectLine": "{{Activity.Field(SubjectLine)}}",
      "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
      "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
      "CampaignId": "{{Activity.Campaign.Id}}",
      "ExternalId": "{{Activity.ExternalId}}",
      "EmailSendType": "{{Activity.Field(EmailSendType)}}"
   },
   "filter": "'{{Activity.Type}}' = 'EmailClickthrough'",
}
```

Export contacts based on their presence in the contact List with ID 123:

```
{
   "name": "List 123 Contacts Export",
   "fields": {
      "ID": "{{Contact.Id}}",
      "FirstName": "{{Contact.Field(C_FirstName)}}",
      "LastName": "{{Contact.Field(C_LastName)}}",
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "filter": "EXISTS('{{ContactList[123]}}')"
}
```

Export contacts based on their presence in the contact segment with ID 5:

```
{
   "name": "Segment 5 Contacts Export",
   "fields": {
      "ID": "{{Contact.Id}}",
      "FirstName": "{{Contact.Field(C_FirstName)}}",
      "LastName": "{{Contact.Field(C_LastName)}}",
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "filter": "EXISTS('{{ContactSegment[5]}}')"
}
```

Note: `EXISTS` operators can only be used on exports of the same entity:

- The ContactList, ContactFilter, or ContactSegment `EXISTS` operators can only be used in filters for contact exports.
- The AccountList `EXISTS` operator can only be used in filters for account exports.

Export contacts for which the AppCloud action service instance ID is f82d50cd86a94fcab37e4ec9a98b0339, with an execution ID of 12345 and a status is *pending*:

```
{
   "name": "Action Service Contacts Export",
   "fields": {
      "ID": "{{Contact.Id}}",
      "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "filter" : "STATUS('{{ActionInstance(f82d50cd86a94fcab37e4ec9a98b0339).Execution
[12345]}}') = 'pending'"
}
```

With `areSystemTimestampsInUTC` request parameter set to `true`, export accounts that have been created after September 9, 2016 13:46:20.975 UTC:

```
{
   "name": "System Timestamps in UTC Export",
   "fields": {
      "companyName": "{{Account.Field(M_CompanyName)}}",
      "createdDate": "{{Account.Field(M_DateCreated)}}",
      "modifiedDate": "{{Account.Field(M_DateModified)}}"
   },
   "areSystemTimestampsInUTC": true,
   "filter": "'{{Account.Field(M_DateCreated)}}' > '2016-09-01 13:46:20.975'"
}
```

Here is an example filtering on the same UTC date value using markers:

```
{
   "name":"System Timestamps in UTC Export",
   "fields":{
      "companyName":"{{Account.Field(M_CompanyName)}}",
      "createdDate":"{{Account.Field(M_DateCreated)}}",
      "modifiedDate":"{{Account.Field(M_DateModified)}}"
   },
   "areSystemTimestampsInUTC":true,
   "filter":"'{{Account.Field(M_DateCreated)}}' > '2016-09-01T13:46:20.975Z'"
}
```

Export all custom object records that are mapped to a contact in custom object with ID 9:

```
{
   "name": "Custom Object Records Export - Mapped to Contacts",
   "fields": {
      "Id": "{{CustomObject[9].ExternalId}}",
      "ContactId": "{{CustomObject[9].Contact.Id}}",
      "ContactFirstName": "{{CustomObject[9].Contact.Field(C_FirstName)}}"
```

```
  },
  "filter": "'{{CustomObject[9].Contact.Id}}' > '0'"
}
```

Export all custom object records that are **not** mapped to a contact in custom object with ID 9:

```
{
  "name": "Custom Object Records Export - Not Mapped to Contacts",
  "fields": {
    "Id": "{{CustomObject[9].ExternalId}}",
    "ContactId": "{{CustomObject[9].Contact.Id}}",
    "ContactFirstName": "{{CustomObject[9].Contact.Field(C_FirstName)}}"
  },
  "filter": "'{{CustomObject[9].Contact.Id}}' = ''"
}
```

Export contact with email address "test.a'pie@test.test":

```
{
  "name": "Exporting Email Address - test.a'pie@test.test",
  "fields": {
    "C_EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "C_FirstName": "{{Contact.Field(C_FirstName)}}"
  },
  "filter": "'{{Contact.Field(C_EmailAddress)}}'='test.a\\'pie@test.test'"
}
```

Export all contacts that are unsubscribed:

```
{
  "name": "Unsubscribed Contacts",
  "fields": {
    "C_EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "IsSubscribed": "{{Contact.Email.IsSubscribed}}"
  },
```

```
  "filter": "'{{Contact.Email.IsSubscribed}}'='0'"
}
```

Export all contacts that are marked as bounced:

```
{
  "name": "Bounced Contacts",
  "fields": {
    "C_EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
    "IsBounced": "{{Contact.Email.IsBounced}}"
  },
  "filter": "'{{Contact.Email.IsBounced}}'='1'"
}
```

Note: When using `{{Contact.Email.IsSubscribed}}` or `{{Contact.Email.IsBounced}}` in an export filter use “0” for False and “1” for True.

Complex filters

The bulk API also supports complex filters that filter based on multiple criteria using the AND, OR, and NOT operators.

Note: You can only use one EXISTS and STATUS operator in a filter.

The bulk API processes complex filters much like a `WHERE` clause in a SQL environment. You can use parentheses to group predicates together.

Important: Supported filter formats with logical operators:

- (A OR B) AND (C OR D)
- A AND NOT B AND (C OR D)
- A AND B AND (C OR D)
- A AND (B OR C)

Activity exports only support the A AND B AND C filter format.

The following export definitions illustrate some complex filters:

Export contacts whose country is Belgium or France:

```
{
   "name": "Belgian and French Contacts Export",
   "fields": {
      "ID": "{{Contact.Id}}",
      "FirstName": "{{Contact.Field(C_FirstName)}}",
      "LastName": "{{Contact.Field(C_LastName)}}",
      "Country": "{{Contact.Field(C_Country)}}"
   },
   "filter": "'{{Contact.Field(C_Country)}}' = 'BE' OR '{{Contact.Field(C_Country)}}' = 'FR'"
}
```

Export activities of type "`EmailClickedThrough`" that occurred in August 2014:

```
{
   "name":"EmailClickthrough Activity Export - Aug 2014",
   "fields": {
      "ActivityId": "{{Activity.Id}}",
      "ActivityType": "{{Activity.Type}}",
```

```
      "ActivityDate": "{{Activity.CreatedAt}}",
      "EmailAddress": "{{Activity.Field(EmailAddress)}}",
      "ContactId": "{{Activity.Contact.Id}}",
      "IpAddress": "{{Activity.Field(IpAddress)}}",
      "VisitorId": "{{Activity.Visitor.Id}}",
      "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
      "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
      "AssetType": "{{Activity.Asset.Type}}",
      "AssetName": "{{Activity.Asset.Name}}",
      "AssetId": "{{Activity.Asset.Id}}",
      "SubjectLine": "{{Activity.Field(SubjectLine)}}",
      "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
      "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
      "CampaignId": "{{Activity.Campaign.Id}}",
      "ExternalId": "{{Activity.ExternalId}}",
      "EmailSendType": "{{Activity.Field(EmailSendType)}}"
   },
   "filter":"'{{Activity.Type}}' = 'EmailClickthrough' AND '{{Activity.CreatedAt}}' >= '2014-08-
01' AND '{{Activity.CreatedAt}}' < '2014-09-01'"
}
```

Export contacts whose `CreatedAt` field is greater than 2013-12-31 and whose account's company name is not Oracle and whose `C_Country` field is Canada or United States:

```
{
   "name": "Contacts Created after 2013 - Not Oracle - Canada or US",
   "fields": {
      "firstName": "{{Contact.Field(C_FirstName)}}",
      "lastName": "{{Contact.Field(C_LastName)}}",
      "email": "{{Contact.Field(C_EmailAddress)}}",
      "country": "{{Contact.Field(C_Country)}}"
   },
   "filter": "'{{Contact.CreatedAt}}' > '2013-12-31' AND '{{Contact.Account.Field(M_
CompanyName)}}' != 'Oracle' AND ('{{Contact.Field(C_Country)}}' = 'CA' OR '{{Contact.Field
(C_Country)}}' = 'US')"
}
```

Export email send activities for a specific range of campaign IDs between 70 and 128:

```
{
  "name": "Bulk Activity Export - Email Send - Campaign Ids between 70 and 128",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  },
  "filter": "'{{Activity.Type}}'='EmailSend' AND '{{Activity.Campaign.Id}}'>='70' AND '
{{Activity.Campaign.Id}}'<='128'"
}
```

Retrieving large volumes of data

Retrieving large volumes of data from Eloqua requires multiple requests using the `offset` and `limit` query parameters. This tutorial will walk you through how to retrieve data surpassing the limit of 50,000 per request, including example HTTP requests and cURL.

In this tutorial:

- Understanding the offset and limit query parameters
- Example requests

Understanding the offset and limit query parameters

The query parameters `limit` and `offset` are used to specify how many records to return, and which batch of records to return. When exporting large amounts of data surpassing 50,000, these query parameters are required to retrieve the next batch of records since the maximum amount of records you can retrieve at once within a single batch is 50,000.

Query parameter	Description	Default	Maximum	Example
`limit`	A URL parameter that specifies the maximum number of records to return. This can be any positive integer between 1 and 50000 inclusive. If not specified, the default is 1000. **Note**: Only the /data endpoints have a maximum limit of 50000.	1000	50000	`?limit=50000`

Query parameter	Description	Default	Maximum	Example
	Refer to the documentation for specific endpoints to learn more about limitations.			
`offset`	Specifies an offset that allows you to retrieve the next batch of records. Any positive integer. For example, if your limit is 1000, specifying an offset of 1000 will return records 1000 through 2000. If not specified, the default is 0.	0	N/A	`?offset=1000`

Note the following:

- It is best practice to export data using the syncs data endpoint.
- By default, data is returned in JSON format unless an Accept header is added to accept text/csv format.

For example, you have 150,000 records in your database. The maximum amount of records you can retrieve at once is 50,000, so you will need to make three separate calls to retrieve 50,000 records at a time. The following examples will walkthrough how to form these requests.

> **Note**: The example requests below assume that our Eloqua pod is 3 and our sync id is 8.

Retrieving records 0 - 50000

To return records 0 to 50,000, we'll specify an `offset` of 0 and a limit of 50000.

HTTP Request

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data?limit=50000&offset=0
```

cURL request

```
curl --user "APITest\API.User" --request GET
https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data?limit=50000&offset=0
```

Response

```
{
  "totalResults": 150000,
  "limit": 50000,
  "offset": 0,
```

```
  "count": 50000,
  "hasMore": true,
  "items": [
    ...
  ],

}
```

The response displays items 0 to 50,000.

Retrieving records 50,000 - 100,000

To return records 50,000 to 100,000, we'll specify an `offset` of 50000 and a limit of 50000.

HTTP request

```
GET
https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data?limit=50000&offset=50000
```

cURL request

```
curl --user "APITest\API.User" --request GET
https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data?limit=50000&offset=50000
```

Response

```
{
  "totalResults": 150000,
  "limit": 50000,
  "offset": 50000,
  "count": 50000,
  "hasMore": true,
  "items": [
```

```
    ...
  ],

}
```

The response displays items 50,000 to 100,000.

Retrieving records 100,000 - 150,000

To return records 100,000 to 150,000, we'll specify an `offset` of 100000 and a limit of 50000.

HTTP request

```
GET
https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data
?limit=50000&offset=100000
```

cURL request

```
curl --user "APITest\API.User" --request GET
https://secure.p03.eloqua.com/api/bulk/2.0/syncs/8/data
?limit=50000&offset=100000
```

Response

```
{
  "totalResults": 150000,
  "limit": 50000,
  "offset": 100000,
  "count": 50000,
  "hasMore": false,
  "items": [
    ...
```

```
  ],
}
```

The response displays items 100,000 to 150,000.

Exporting activities

The following steps show how to export activities from Eloqua. The same process can be used for any activity export by changing the filter and fields definitions. See our Activities export definitions page for a complete list.

All exports follow the same basic flow:

1. Define the export
2. Create the sync
3. Retrieve the data

To export activities using the Bulk API:

1. Define the export

In order to export email open activities, first create an export definition. This defines the data to be exported including any required filters. For this example we will use the email open activity to demonstrate the process.

Important:

- There is a maximum of 5 million records per Activity export sync. If necessary, you'll need to use a date/time filter to reduce the number of records exported to below 5 million. Activity volume can be checked using Insight reports. No filter is needed if there are fewer than 5 million records. Learn more about Bulk filtering.
- There is a maximum of 10 contact fields allowed in an activity export definition. The addition of contact fields to activity exports will add to export time.
- Contact fields cannot be used in the filter.
- Bouncebacks are recorded asynchronously, which means the Bulk API Activity Created Date (when the bounceback occurred), may not be available to export until hours or days after the bounceback occurred. The upper date limit for a filter should be at least one hour before the time of Bulk API sync creation to ensure the majority of bouncebacks have been recorded.

This example uses a filter to export one month of data: in this case April 2015. Be sure to filter to a smaller time-frame if you think this may still return more than 5 million records:

```
"filter": "'{{Activity.Type}}'='EmailOpen' AND '{{Activity.CreatedAt}}'>=
'2015-04-01' AND '{{Activity.CreatedAt}}' < '2015-05-01'"
```

Once you created an appropriate filter, create a new activity export definition. For this example, we will include all possible email open activity fields, however you only need to use fields whose values you need to export.

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='EmailOpen'",
  "name": "Bulk Activity Export - Email Open",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  }
}
```

When the above export definition is POSTed to `/activities/exports` an export definition is created. Eloqua responds with the following:

Response body:

```
{
  "name": "Bulk Activity Export - Email Open",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
```

```
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  },
  "filter": "'{{Activity.Type}}'='EmailOpen'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/1560",
  "createdBy": "API.User",
  "createdAt": "2017-08-09T16:22:09.0434041Z",
  "updatedBy": "API.User",
  "updatedAt": "2017-08-09T16:22:09.0434041Z"
}
```

In the above response the URI `/activities/exports/1560` is used to trigger the export sync. Be sure to save this value for the following steps.

2. Create the sync

Using the URI value from step 1 (in this case `/activities/exports/1560`) we can now create a sync. To do so, submit a POST request with the URI passed as the `syncedInstanceUri` value:

Request:

```
POST /syncs
```

Request body:

```
{
  "syncedInstanceUri": "/activities/exports/1560"
}
```

A response is then received and the sync is now queued to be automatically run.

Response body:

```
{
  "syncedInstanceUri": "/activities/exports/1560",
  "status": "pending",
  "createdAt": "2017-08-09T16:22:09.0434041Z",
  "createdBy": "API.User",
  "uri": "/syncs/5744"
}
```

The status of the export can be checked by performing a GET on the URI of the sync.

In this case, `/syncs/5744`.

When the export is complete, the status value will be "success":

```
{
  "syncedInstanceUri": "/activities/exports/1560",
  "syncStartedAt": "2017-08-24T20:54:03.8330000Z",
  "syncEndedAt": "2017-08-24T20:54:05.6630000Z",
  "status": "success",
  "createdAt": "2017-08-09T16:22:09.0434041Z",
  "createdBy": "API.User",
  "uri": "/syncs/5744"
}
```

3. Retrieve the data

The data can now be retrieved by performing a GET on the URI's `/data` endpoint, in this case `/syncs/5744/data`.

Request:

```
GET /syncs/5744/data
```

By default this will return the first 1,000 records exported. In order to refine your data selection learn more about URL parameters.

Learn more

Activities - export definitions

Activities export definitions

Below are example export definition requests for all activity types. Creating an export definition is the first step in exporting records from Eloqua. The examples use all possible export fields for each activity type, but you only need to use the fields you wish to actually export and use. If you expect more than 5,000,000 records to be returned in any given export, be sure to filter the results further. Learn more about exporting activities.

Activity types:

- Bounceback
- Email clickthrough
- Email open
- Email send
- External activities

- Form submit
- Page view
- Subscribe
- Unsubscribe
- Web visit

> **Important**: There is a maximum of 10 contact fields allowed in an activity export definition. The addition of contact fields to activity exports will add to export time. Contact fields cannot be used in the filter.

Bounceback

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='Bounceback'",
  "name": "Bulk Activity Export - Bounceback",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
```

```
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
    "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
    "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Bounceback",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
    "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
    "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
  },
  "filter": "'{{Activity.Type}}'='Bounceback'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/73",
  "createdBy": "API.User",
  "createdAt": "2018-09-20T13:36:51.9304577Z",
```

```
  "updatedBy": "API.User",
  "updatedAt": "2018-09-20T13:36:51.9304577Z"
}
```

Email clickthrough

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='EmailClickthrough'",
  "name": "Bulk Activity Export - Email Clickthrough",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
```

```
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Email Clickthrough",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  },
  "filter": "'{{Activity.Type}}'='EmailClickthrough'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/65",
  "createdBy": "API.User",
```

```
  "createdAt": "2018-07-17T14:42:39.0510345Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-07-17T14:42:39.0510345Z"
}
```

Email open

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='EmailOpen'",
  "name": "Bulk Activity Export - Email Open",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
```

```
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  }
}
```

Response body:

```
{
    "name": "Bulk Activity Export - Email Open",
    "fields": {
        "ActivityId": "{{Activity.Id}}",
        "ActivityType": "{{Activity.Type}}",
        "ActivityDate": "{{Activity.CreatedAt}}",
        "ContactId": "{{Activity.Contact.Id}}",
        "IpAddress": "{{Activity.Field(IpAddress)}}",
        "VisitorId": "{{Activity.Visitor.Id}}",
        "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
        "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
        "AssetType": "{{Activity.Asset.Type}}",
        "AssetName": "{{Activity.Asset.Name}}",
        "AssetId": "{{Activity.Asset.Id}}",
        "SubjectLine": "{{Activity.Field(SubjectLine)}}",
        "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
        "CampaignId": "{{Activity.Campaign.Id}}",
        "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
        "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
        "ExternalId": "{{Activity.ExternalId}}",
        "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
        "EmailSendType": "{{Activity.Field(EmailSendType)}}",
        "EmailAddress": "{{Activity.Field(EmailAddress)}}",
        "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
        "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
        "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
    },
    "filter": "'{{Activity.Type}}'='EmailOpen'",
    "dataRetentionDuration": "PT12H",
    "uri": "/activities/exports/66",
    "createdBy": "API.User",
    "createdAt": "2018-07-17T14:50:52.3183789Z",
```

```
    "updatedBy": "API.User",
    "updatedAt": "2018-07-17T14:50:52.3183789Z"
}
```

Email send

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='EmailSend'",
  "name": "Bulk Activity Export - Email Send",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  }
}
```

Response body:


```
{
  "name": "Bulk Activity Export - Email Send",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "SubjectLine": "{{Activity.Field(SubjectLine)}}",
    "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  },
  "filter": "'{{Activity.Type}}'='EmailSend'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/67",
  "createdBy": "API.User",
  "createdAt": "2018-07-17T17:46:28.9274083Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-07-17T17:46:28.9274083Z"
}
```

External activities

Request:

```
POST /activities/exports
```

Request body:

```
{
   "filter": "'{{Activity.Type}}' = 'ExternalActivity'",
   "name": "Bulk Activity Export - External Activity",
   "fields": {
      "ActivityId": "{{Activity.Id}}",
      "ActivityType": "{{Activity.Type}}",
      "ActivityDate": "{{Activity.CreatedAt}}",
      "ExternalAssetName": "{{Activity.ExternalAssetName}}",
      "ExternalAssetType": "{{Activity.ExternalAssetType}}",
      "ExternalType": "{{Activity.ExternalActivityType}}",
      "CampaignId": "{{Activity.Campaign.Id}}",
      "ContactId": "{{Activity.Contact.Id}}",
      "ExternalId": "{{Activity.ExternalId}}",
      "EmailAddress": "{{Activity.Field(EmailAddress)}}",
      "ContactId": "{{Activity.Contact.Id}}",
      "CampaignId": "{{Activity.Campaign.Id}}",
      "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
      "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
      "CampaignResponseCreatedAt": "{{Activity.CampaignResponse.CreatedAt}}",
      "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}",
      "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
   }
}
```

Response body:

```
{
   "name": "Bulk Activity Export - External Activity",
   "fields": {
      "ActivityId": "{{Activity.Id}}",
      "ActivityType": "{{Activity.Type}}",
      "ActivityDate": "{{Activity.CreatedAt}}",
```

```
      "ExternalAssetName": "{{Activity.ExternalAssetName}}",
      "ExternalAssetType": "{{Activity.ExternalAssetType}}",
      "ExternalType": "{{Activity.ExternalActivityType}}",
      "CampaignId": "{{Activity.Campaign.Id}}",
      "ContactId": "{{Activity.Contact.Id}}",
      "ExternalId": "{{Activity.ExternalId}}",
      "EmailAddress": "{{Activity.Field(EmailAddress)}}",
      "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
      "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
      "CampaignResponseCreatedAt": "{{Activity.CampaignResponse.CreatedAt}}",
      "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}",
      "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
   },
   "filter": "'{{Activity.Type}}' = 'ExternalActivity'",
   "maxRecords": 5000000,
   "dataRetentionDuration": "PT12H",
   "uri": "/activities/exports/425",
   "createdBy": "API.User",
   "createdAt": "2023-07-31T23:04:26.4070000Z",
   "updatedBy": "API.User",
   "updatedAt": "2023-07-31T23:04:26.4070000Z"
}
```

Form submit

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='FormSubmit'",
  "name": "Bulk Activity Export - Form Submit",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
```

```
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "RawData": "{{Activity.Field(RawData)}}",
    "FormSubmitSavedId": "{{Activity.Field(FormSubmitSavedId)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Form Submit",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "RawData": "{{Activity.Field(RawData)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
```

```
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  },
  "filter": "'{{Activity.Type}}'='FormSubmit'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/68",
  "createdBy": "API.User",
  "createdAt": "2018-07-17T17:51:21.6851479Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-07-17T17:51:21.6851479Z"
}
```

Page view

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='PageView'",
  "name": "Bulk Activity Export - Page View",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "WebVisitId": "{{Activity.Field(WebVisitId)}}",
    "Url": "{{Activity.Field(Url)}}",
    "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "IsWebTrackingOptedIn": "{{Activity.Field(IsWebTrackingOptedIn)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
```

```
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
    "PageViewSavedId": "{{Activity.Field(PageViewSavedId)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Page View",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "WebVisitId": "{{Activity.Field(WebVisitId)}}",
    "Url": "{{Activity.Field(Url)}}",
    "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "IsWebTrackingOptedIn": "{{Activity.Field(IsWebTrackingOptedIn)}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
    "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
    "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
  },
  "filter": "'{{Activity.Type}}'='PageView'",
```

```
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/69",
  "createdBy": "API.User",
  "createdAt": "2018-07-17T17:54:06.3563512Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-07-17T17:54:06.3563512Z"
}
```

Subscribe

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='Subscribe'",
  "name": "Bulk Activity Export - Subscribe",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Subscribe",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  },
  "filter": "'{{Activity.Type}}'='Subscribe'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/74",
  "createdBy": "API.User",
  "createdAt": "2018-09-20T13:38:55.5560961Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-09-20T13:38:55.5560961Z"
}
```

Unsubscribe

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='Unsubscribe'",
```

```
  "name": "Bulk Activity Export - Unsubscribe",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  }
}
```

Response body:

```
{
  "name": "Bulk Activity Export - Unsubscribe",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "AssetType": "{{Activity.Asset.Type}}",
    "AssetName": "{{Activity.Asset.Name}}",
    "AssetId": "{{Activity.Asset.Id}}",
    "CampaignId": "{{Activity.Campaign.Id}}",
    "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
    "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
  },
  "filter": "'{{Activity.Type}}'='Unsubscribe'",
```

```
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/75",
  "createdBy": "API.User",
  "createdAt": "2018-09-20T13:42:03.4178334Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-09-20T13:42:03.4178334Z"
}
```

Web visit

Request:

```
POST /activities/exports
```

Request body:

```
{
  "filter": "'{{Activity.Type}}'='WebVisit'",
  "name": "Bulk Activity Export - Web Visit",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "NumberOfPages": "{{Activity.Field(NumberOfPages)}}",
    "FirstPageViewUrl": "{{Activity.Field(FirstPageViewUrl)}}",
    "Duration": "{{Activity.Field(Duration)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
    "WebVisitSavedId": "{{Activity.Field(WebVisitSavedId)}}"
  }
}
```

Response body:


```
{
  "name": "Bulk Activity Export - Web Visit",
  "fields": {
    "ActivityId": "{{Activity.Id}}",
    "ActivityType": "{{Activity.Type}}",
    "ActivityDate": "{{Activity.CreatedAt}}",
    "ContactId": "{{Activity.Contact.Id}}",
    "VisitorId": "{{Activity.Visitor.Id}}",
    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
    "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
    "IpAddress": "{{Activity.Field(IpAddress)}}",
    "NumberOfPages": "{{Activity.Field(NumberOfPages)}}",
    "FirstPageViewUrl": "{{Activity.Field(FirstPageViewUrl)}}",
    "Duration": "{{Activity.Field(Duration)}}",
    "ExternalId": "{{Activity.ExternalId}}",
    "EmailAddress": "{{Activity.Contact.Field(C_EmailAddress)}}",
    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}",
    "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
    "WebVisitSavedId": "{{Activity.Field(WebVisitSavedId)}}"
  },
  "filter": "'{{Activity.Type}}'='WebVisit'",
  "dataRetentionDuration": "PT12H",
  "uri": "/activities/exports/39",
  "createdBy": "API.User",
  "createdAt": "2017-08-09T16:14:57.0142347Z",
  "updatedBy": "API.User",
  "updatedAt": "2017-08-09T16:14:57.0142347Z"
}
```


Activity fields

Activity elements behave differently from other Eloqua elements. Each activity type has its own set of associated fields which are detailed below.

- EmailOpen
- EmailClickthrough
- EmailSend

- Subscribe
- Unsubscribe
- Bounceback
- WebVisit
- PageView
- FormSubmit
- ExternalActivity

For activity field type and length, see activity field details.

EmailOpen activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}",
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

```
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

EmailClickthrough activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}"
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

EmailSend activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}"
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

Subscribe activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

```
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}"
}
```

Unsubscribe activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}"
}
```

Bounceback activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
  "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
  "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
}
```

WebVisit activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "NumberOfPages": "{{Activity.Field(NumberOfPages)}}",
  "FirstPageViewUrl": "{{Activity.Field(FirstPageViewUrl)}}",
  "Duration": "{{Activity.Field(Duration)}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
  "WebVisitSavedId": "{{Activity.Field(WebVisitSavedId)}}"
}
```

PageView activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "WebVisitId": "{{Activity.Field(WebVisitId)}}",
  "Url": "{{Activity.Field(Url)}}",
  "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "IsWebTrackingOptedIn": "{{Activity.Field(IsWebTrackingOptedIn)}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
  "PageViewSavedId": "{{Activity.Field(PageViewSavedId)}}",
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

FormSubmit activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "RawData": "{{Activity.Field(RawData)}}",
  "FormSubmitSavedId": "{{Activity.Field(FormSubmitSavedId)}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

ExternalActivity activity fields

```
{
   "ActivityId": "{{Activity.Id}}",
   "ActivityType": "{{Activity.Type}}",
   "ActivityDate": "{{Activity.CreatedAt}}",
   "ExternalAssetName": "{{Activity.ExternalAssetName}}",
   "ExternalAssetType": "{{Activity.ExternalAssetType}}",
   "ExternalType": "{{Activity.ExternalActivityType}}",
   "CampaignId": "{{Activity.Campaign.Id}}",
   "ContactId": "{{Activity.Contact.Id}}",
   "ExternalId": "{{Activity.ExternalId}}",
   "EmailAddress": "{{Activity.Field(EmailAddress)}}",
   "ContactId": "{{Activity.Contact.Id}}",
   "CampaignId": "{{Activity.Campaign.Id}}",
   "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
   "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
   "CampaignResponseCreatedAt": "{{Activity.CampaignResponse.CreatedAt}}",
   "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
```

```
}
```

Activity field details

Field type	Data type	Max length	Description
`{{Activity.Id}}`	Integer (big Integer for page views)	-	The unique identifier of the activity per activity type, except for EmailSend The activity ID is not unique across activity types, or for EmailSend.
`{{Activity.Type}}`	String	100	The type of activity. The activity types are EmailOpen, EmailClickthrough, EmailSend, Subscribe, Unsubscribe, Bounceback, WebVisit, PageView, and FormSubmit.
`{{Activity.CreatedAt}}`	Date/time stamp	-	The date and time of the activity
`{{Activity.Field(EmailAddress)}}`	String	400	The email address of the contact who performed the activity
`{{Activity.Contact.Id}}`	Integer	-	The ID of the contact who performed the activity
`{{Activity.Field(EmailRecipientId)}}`	String	38	The recipient ID associated with an email activity

Field type	Data type	Max length	Description
			The recipient ID is present on every link in an email in the format of elq=*GUID*. At the time of sending the email, a GUID is generated and injected into the link. Learn more about the RecipientId.
`{{Activity.Asset.Type}}`	String	100	The type of asset associated with the activity, such as email and form
`{{Activity.Asset.Name}}`	String	100	The name of the asset associated with the activity
`{{Activity.Asset.Id}}`	Integer	-	The ID of the asset associated with the activity
`{{Activity.Field(SubjectLine)}}`	String	500	The subject line for an email activity
`{{Activity.Field(EmailWebLink)}}`	String	1000	The link for viewing the email in a web browser window for an email activity
`{{Activity.Campaign.Id}}`	Integer	-	The ID of the campaign associated with the

Field type	Data type	Max length	Description
			activity
{{Activity.Campaign.Field (CRMCampaignId)}}	String	100	The ID of the CRM campaign associated with the activity
{{Activity.Campaign.Field (CampaignName)}}	String	100	The name of the campaign associated with the activity
{{Activity.Field (EmailDeploymentId)}}	Integer	-	The ID of the email deployment for an email activity
{{Activity.Field(IpAddress)}}	String	50	The IP address of the visitor who performed the activity
{{Activity.Visitor.Id}}	Integer	-	The ID of the visitor who performed the activity
{{Activity.Visitor.ExternalId}}	String	38	The GUID of the visitor who performed the activity
{{Activity.Field (EmailClickedThruLink)}}	String	1000	The link clicked through for an EmailClickthrough activity
{{Activity.Field(RawData)}}	String	64000	The raw data submitted with a FormSubmit activity. Includes the HTML field name and the submitted values in a query string format.
{{Activity.Field(ReferrerUrl)}}	String	1000	The URL or form submission a PageView or WebVisit activity was referred from. Example form submission:

Field type	Data type	Max length	Description
			"Form Submitted: <FormName>"
{{Activity.Field(WebVisitId)}}	Integer	-	The ID of the WebVisit associated with a PageView activity
{{Activity.Field(Url)}}	String	1000	The URL viewed with a PageView activity
{{Activity.Field (IsWebTrackingOptedIn)}}	Boolean	-	Indicates if visitor who performed activity is opted-in to web tracking
{{Activity.Field(NumberOfPages)}}	Integer	-	The number of pages viewed with a WebVisit activity
{{Activity.Field(FirstPageViewUrl)}}	String	1000	The URL of the first page viewed in a WebVisit activity
{{Activity.Field(Duration)}}	String (ISO-8601 Duration)	100	The duration of a WebVisit activity returned using the ISO-8601 standard for specifying durations
{{Activity.Field(EmailSendType)}}	String	100	The type of email send for an email activity. Email send types are Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI,

Field type	Data type	Max length	Description
			SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, and SecureEmailDeployment.
{{Activity.ExternalId}}	String	20	The unique identifier of the activity across activity types
{{Activity.Field(SmtpErrorCode)}}	String	9	The SMTP Status Code for the email bounceback
{{Activity.Field(SmtpStatusCode)}}	String	3	The SMTP Response Code for the email bounceback
{{Activity.Field(SmtpMessage)}}	String	510	The SMTP message for the email bounceback
{{Activity.Field (LinkedToContactDate)}}	Date and time stamp	-	The date and time the visitor was linked to the contact
{{Activity.Field(WebVisitSavedId)}}	Integer	--	The unique identifier of the web visit session.
{{Activity.Field(PageViewSavedId)}}	Integer	--	The unique identifier of the page view.
{{Activity.Field (FormSubmitSavedId)}}	Integer	--	The unique identifier of the form submission.
{{Activity.CampaignResponse.CreatedAt}}	Date/time stamp	-	The date and time the campaign response was created.
{{Activity.CampaignResponse.Field (MemberStatus)}}	String	150	The status of the campaign member.

Field type	Data type	Max length	Description
`{{Activity.ExternalAssetName}}`	String	100	The first sub-category for the asset type. For example, if it is a Tradeshow asset type, then the asset name could be the name of the Tradeshow.
`{{Activity.ExternalAssetType}}`	String	100	The top-level classification for something that your contacts are performing, for example, a Tradeshow.
`{{Activity.ExternalActivityType}}`	String	100	The specific activity that the contact or prospect performed for this specific external asset. For example, if it is a tradeshow event, then activities could include "Registered", "Canceled", "Completed", and so on.

Using the campaign response endpoints

The campaign response endpoints enable App Developers to send campaign responses from Eloqua to a CRM, using an Action Service on a Program or Campaign Canvas.

Feature highlights

- Enables sending campaign responses to any CRM using an App Action Service on a Program or Campaign Canvas. Previously, campaign responses could only be sent using Integration Rules or Program Builder to a native CRM integration.
- Limits use to one app service instance to ensure all campaign responses are successfully sent to the designated CRM.

Use cases

The usage of the campaign response endpoints are tied to a single instance of an app service. During the setup, users must select the app, the service, and the exact app service instance that can export and import campaign responses. This is done to prevent other users from interacting with campaign responses, except for the app service instance selected.

Workflow

The process of leveraging the campaign response endpoints is as follows:

Task	Responsible Role
Develop an Action service specifically for campaign responses	App Developers
Share the install URL with users	App Developers
Install the app and service	Marketers
Drag the service into a Program or Campaign Canvas	Marketers
Configure response rules	Marketers
Activate the campaign or program to start receiving notifications	Marketers

See the following topics to get started.

Tasks

Developing for campaign responses

Campaign response setup for marketers

API Reference

Campaign Responses API

Developing for campaign responses

This tutorial provides a walkthrough of considerations when developing an app and service for Campaign Responses. The campaign response endpoints generally work the same as other Bulk API endpoints, however there are some unique considerations. The following guidelines are meant to instruct developers on how to best use the campaign response endpoints. In this tutorial:

- Understanding workflows
- Developing apps and services
- Calling the endpoints
- Troubleshooting

Understanding workflows

As an app provider, your implementation of your app workflow may be unique. Here is an example workflow of how developers could use these endpoints:

1. Contacts enter app service step and Eloqua notifies app.
2. App retrieves campaign responses from Eloqua with a campaign response definition.
3. App sends campaign responses to a CRM.

4. App responds back to Eloqua with a campaign response import definition for every successful campaign response created in the CRM.
5. App uses a sync action to move contacts along.

Developing apps and services

The following are guidelines to ensure that apps and services that are used to send campaign responses to CRM are optimized correctly:

- **Ensure Records per Notification is set to 0**

  During Action Service registration, we recommend that the **Records per Notification** should be is set to 0 given you are always going to be exporting campaign responses in a separate sync.

Calling the endpoints

POST `/api/bulk/2.0/campaignResponses/exports`

API reference

General endpoint usage

In the `filter`, your service instance ID is specified. If your service instance ID has not been selected to consume campaign responses, a validation error will be returned.

This export requires records to be in the step. In the `filter`, a valid execution ID is also required, and the validation error will be returned if there is not a valid execution ID.

```
{
  "name": "Example Campaign Response Export",
  "fields": {
    "CampaignResponseId": "{{CampaignResponse.Id}}",
    "MemberStatus": "{{CampaignResponse.Field(MemberStatus)}}",
```

```
    "CampaignMembershipId": "{{CampaignResponse.Field(IntegrationReturnValue)}}",
    "CreatedAt": "{{CampaignResponse.CreatedAt}}",
    "CRMCampaignId": "{{CampaignResponse.Campaign.Field(CRMCampaignId)}}",
    "EloquaCampaignId": "{{CampaignResponse.Campaign.Id}}",
    "EloquaCampaignName": "{{CampaignResponse.Campaign.Field(CampaignName)}}",
    "LeadId": "{{CampaignResponse.Contact.Field(C_OracleFusionLeadID)}}",
    "ContactId": "{{CampaignResponse.Contact.Field(C_OracleFusionContactID)}}",
    "EmailAddress": "{{CampaignResponse.Contact.Field(C_EmailAddress)}}"
  },
  "filter": "STATUS('{{ActionInstance(7BE704D3AF604775AD416E8D0A5AB212).Execution
[12345]}}') = 'pending'"
}
```

Date ranges

By default, when creating campaign response export definitions, you will get 30 days of campaign responses. This is the limit of how far in the past campaign responses can be retrieved.

If you want to retrieve anything other than 30 days, developers can use a date range in the `filter` to filter results to be anything **less** than 30 days. If a date filter is **greater** than 30 days, a validation error will be returned.

Here's an example of a campaign response export definition using a date range filter.

```
{
  "name": "Example Campaign Response Export with date filter",
  "fields": {
    "CampaignResponseId": "{{CampaignResponse.Id}}",
    "MemberStatus": "{{CampaignResponse.Field(MemberStatus)}}",
    "CampaignMembershipId": "{{CampaignResponse.Field(IntegrationReturnValue)}}",
    "CreatedAt": "{{CampaignResponse.CreatedAt}}",
    "CRMCampaignId": "{{CampaignResponse.Campaign.Field(CRMCampaignId)}}",
    "EloquaCampaignId": "{{CampaignResponse.Campaign.Id}}",
    "EloquaCampaignName": "{{CampaignResponse.Campaign.Field(CampaignName)}}",
    "LeadId": "{{CampaignResponse.Contact.Field(C_OracleFusionLeadID)}}",
```

```
    "ContactId": "{{CampaignResponse.Contact.Field(C_OracleFusionContactID)}}",
    "EmailAddress": "{{CampaignResponse.Contact.Field(C_EmailAddress)}}"
  },
  "filter": "STATUS('{{ActionInstance(7BE704D3AF604775AD416E8D0A5AB212).Execution
[12345]}}') = 'pending' AND '{{CampaignResponse.CreatedAt}}' > '2017-09-01 13:41:20.985'"
}
```

POST `/api/bulk/2.0/campaignResponses/imports`

API reference

General endpoint usage

You must specify your service instance id with the `serviceInstanceId` property and this must match the selected service instance to consume campaign responses, or else a validation error will be returned. We recommend using the `CampaignMembershipId` from the CRM for the `IntegrationReturnValue` Campaign Response field. The existence of the `IntegrationReturnValue` Campaign Response field value is how we know not to send this Campaign Response record in the future, so a value should be imported to this field for every campaign response record that is successfully created in CRM.

Both the `CampaignResponseId` and `IntegrationReturnValue` fields are required in a campaign response import. Additionally, these are the only two fields allowed in a campaign response import.

This endpoint could be used to test if your instance has been selected to consume campaign responses, and block activating a canvas if it has not. For example, the `requiresConfiguration` property on the DTO could be set to false, requiring the User to click the instance configuration. Within configuration, there could be a **Validate instance selected to consume campaign responses** button, which could initiate a

campaign responses import with the service instance ID to confirm if it is successful. If successful, the app could set `requiresConfiguration` to true and unblock activation. If not successful, instructions could be provided on how to set the instance to consume campaign responses.

It is important to note that there are various campaign responses that could exist for a contact. In the scenario where there are multiple campaign responses for a contact and one of those responses errors, app providers must determine how to handle that error.

Troubleshooting

If you experience any difficulty using the campaign response endpoints, see the following troubleshooting tips.

Ensure the selected service instance within a Campaign or Program canvas is in a draft or deactivated state before changing the selection

If the selected service instance is in an Active canvas, the selection cannot be changed. To change the instance to consume campaign responses, the canvas the current selection is located must be deactivated.

What happens if contacts are reaching the step, but do not have any campaign responses?

If contacts reach the step, but do not have any campaign responses, records with contact fields are sent and all campaign response fields are blank for these records.

Campaign response setup for marketers

This topic instructs marketers on how to leverage the campaign response functionality within their Eloqua instance. This process involves four steps:

1. Install an app developed for sending campaign responses to a CRM.
2. Drag the app service instance into a campaign or program canvas designed for campaign responses.
3. Configure your campaign responses within Eloqua to use the app, service, and campaign or program you have orchestrated.
4. Activate the campaign or program.

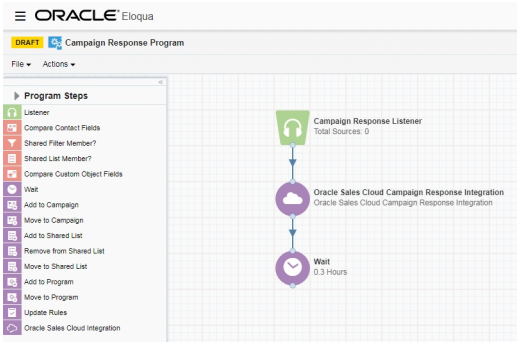
Important: Avoid uploading contact activities that are older than 30 days. These contact activities will not be sent to apps.

Step 1 - Installing the app

Contact the App Provider for the app install URL. The App Provider should be able to guide you through the installation.

Step 2 - Orchestrate a campaign or program using the app

Create a campaign or program to be used for sending campaign responses to your CRM. Here's an example of a sample contact program, where our service is called *Oracle Sales Cloud Campaign Response Integration*.

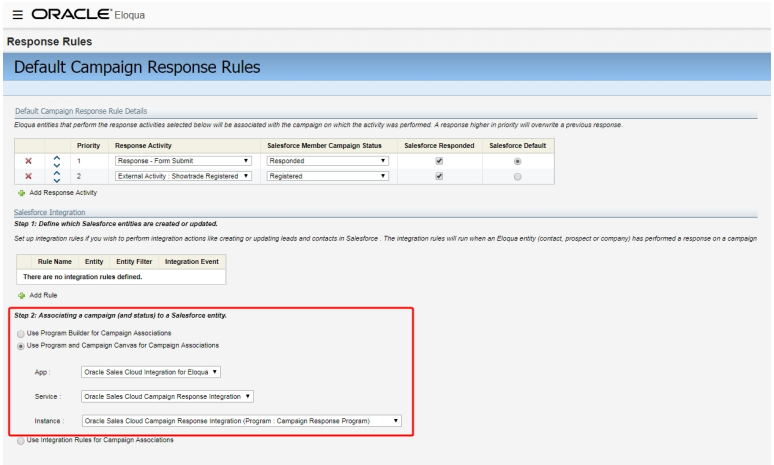


Keep the campaign or program in the draft stage for now, we will activate it during step 4. If you need help creating a campaign or program, see the Oracle Eloqua Help Center.

> **Note**: This is an example of a standalone campaign response Program. This step is also commonly added to the larger CRM integration Program. Regardless of where the step is placed, a Campaign Responses Listener is typically added to feed in contacts that have new campaign responses as they occur.

Step 3 - Configuring campaign responses

1. Navigate to **Settings** > **Response Rules**.
2. Select **Use Program and Campaign Canvas for Campaign Associations**.
3. Click the drop downs to select the App, Service, and Instance you will use to consume campaign responses.



- **App**: Select the app you want to use to send campaign responses to your CRM.
- **Service**: Select the service you want to use to send campaign responses to your CRM.
- **Instance**: Select the campaign or program instance where your app service instance is being used to send campaign responses to your CRM. You created this in step 2.

4. Click **Save**.

> **Note**: To change the instance to consume campaign responses, the canvas the current selection is located must be deactivated.

Step 4 - Activate the campaign or program

Navigate back to your campaign or program, and click **Activate**. You'll then start receiving campaign responses.

Using the opportunities endpoints

The opportunities API endpoints enable developers to import opportunities into Eloqua's opportunity object, and then link the opportunities directly to contacts or via accounts by linking the opportunity to all contacts linked at the indicated account.

The opportunities endpoints are similar to other Bulk API endpoints, the differentiator for opportunities is that you need to upload opportunities, and then do a separate import to link opportunities to contacts. This tutorial will walk through the steps needed to import opportunities and link to contacts. For information about the endpoints, see the reference documentation.

Importing opportunities

Required fields and limitations

The following fields are required in the request body.

- `{{Opportunity.Id}}`
- `{{Opportunity.Field(Name)}}`
- `{{Opportunity.Field(Stage)}}`
- `{{Opportunity.Field(Amount)}}`
- `identifierFieldName`

  Must contain the `{{Opportunity.Id}}` field within `fields`.

To import opportunities

1. Create a new opportunity import definition.

   ```
   POST /api/bulk/2.0/opportunities/imports
   Content-Type: application/json
   ```

   Request body

   ```
   {
     "name": "Opportunity Import",
     "fields": {
       "OpportunityID": "{{Opportunity.Id}}",
   ```

```
    "OpportunityName": "{{Opportunity.Field(Name)}}",
    "AccountName": "{{Opportunity.Field(AccountName)}}",
    "CreatedDate": "{{Opportunity.CreatedAt}}",
    "Amount": "{{Opportunity.Field(Amount)}}",
    "CloseDate": "{{Opportunity.Field(CloseDate)}}",
    "Currency": "{{Opportunity.Field(Currency)}}",
    "ForecastToCloseDate": "{{Opportunity.Field(ForecastToCloseDate)}}",
    "Stage": "{{Opportunity.Field(Stage)}}",
    "Territory": "{{Opportunity.Field(Territory)}}",
    "Owner": "{{Opportunity.Field(Owner)}}",
    "PrimaryCampaignId": "{{Opportunity.Field(PrimaryCampaignId)}}"
  },
  "identifierFieldName": "OpportunityID",
  "isIdentifierFieldCaseSensitive": false
}
```

Response

```
{
    "isIdentifierFieldCaseSensitive": false,
    "name": "Opportunity Import",
    "fields": {
        "OpportunityID": "{{Opportunity.Id}}",
        "OpportunityName": "{{Opportunity.Field(Name)}}",
        "AccountName": "{{Opportunity.Field(AccountName)}}",
        "CreatedDate": "{{Opportunity.CreatedAt}}",
        "Amount": "{{Opportunity.Field(Amount)}}",
        "CloseDate": "{{Opportunity.Field(CloseDate)}}",
        "Currency": "{{Opportunity.Field(Currency)}}",
        "ForecastToCloseDate": "{{Opportunity.Field(ForecastToCloseDate)}}",
        "Stage": "{{Opportunity.Field(Stage)}}",
        "Territory": "{{Opportunity.Field(Territory)}}",
        "Owner": "{{Opportunity.Field(Owner)}}",
        "PrimaryCampaignId": "{{Opportunity.Field(PrimaryCampaignId)}}"
    },
    "identifierFieldName": "OpportunityID",
    "isSyncTriggeredOnImport": false,
    "dataRetentionDuration": "P7D",
    "uri": "/opportunities/imports/24",
```

```
  "createdBy": "API.User",
  "createdAt": "2022-06-13T14:02:29.6270000Z",
  "updatedBy": "API.User",
  "updatedAt": "2022-06-13T14:02:29.6270000Z"
}
```

2. Upload data to the definition.

```
POST /api/bulk/2.0/opportunities/imports/24/data
Content-Type: application/json
```

Request body

```
[
 {
  "OpportunityID": "1",
  "OpportunityName": "ABC Company",
  "AccountName": "ABC",
  "CreatedDate": "2017-10-05 1:15",
  "Amount": "1000000",
  "CloseDate": "",
  "Currency": "USD",
  "ForecastToCloseDate": "",
  "Stage": "Prospecting",
  "Territory": "West"
 },
 {
  "OpportunityID": "2",
  "OpportunityName": "XYZ Company",
  "AccountName": "XYZ",
  "CreatedDate": "2017-10-05 1:25",
  "Amount": "1000000",
  "CloseDate": "",
  "Currency": "USD",
  "ForecastToCloseDate": "",
  "Stage": "Prospecting",
  "Territory": "West"
 }
```

```
]
```

Response

```
204 No Content
```

3. Synchronize the data into Eloqua.

```
POST /api/bulk/2.0/syncs
Content-Type: application/json
```

Request body

```
{
  "syncedInstanceUri": "/opportunities/imports/24"
}
```

Response

```
{
  "syncedInstanceUri": "/opportunities/imports/24",
  "status": "pending",
  "createdAt": "2022-06-13T14:02:29.6270000Z",
  "createdBy": "API.User",
  "uri": "/syncs/10"
}
```

When the sync is successful, the response will resemble:

```
{
   "syncedInstanceUri": "/opportunities/imports/24",
   "syncStartedAt": "2022-06-13T14:02:29.6270000Z",
   "status": "success",
   "createdAt": "2022-06-13T14:02:29.6270000Z",
   "createdBy": "API.User",
   "uri": "/syncs/10"
}
```

Linking to contacts

Required fields and limitations

Note the following required fields and limitations when creating an opportunity contact linkage import definition.

- The `{{Opportunity.Id}}` field is required within `fields`.
- At least one contact or account field is required within `fields`.
- There is a maximum of two `fields` allowed.

To link to contacts:

1. Create a new opportunity contact linkage import definition:

```
POST /api/bulk/2.0/opportunities/contacts/imports
Content-Type: application/json
```

Request body

```
{
 "name": "Opportunity Contact Linkage Import",
 "fields": {
  "EmailAddress": "{{Opportunity.Contact.Field(C_EmailAddress)}}",
  "OpportunityID": "{{Opportunity.Id}}"
 },
 "linkOpportunitiesCaseSensitiveMatchField": false,
 "linkOpportunitiesCaseSensitiveSourceField": false,
 "linkOpportunitiesEntityType": "Contact",
 "linkOpportunitiesMatchFieldName": "OpportunityID",
 "linkOpportunitiesMultipleSourceMatches": true,
 "linkOpportunitiesSourceField": "EmailAddress"
}
```

In our import definition, we are stating that we want to link contacts by their email address to specific opportunities, by `OpportunityID`. For more information about the parameters shown here, see the reference documentation.

Response

```
{
 "linkOpportunitiesMatchFieldName": "OpportunityID",
 "linkOpportunitiesSourceField": "EmailAddress",
 "linkOpportunitiesEntityType": "Contact",
 "linkOpportunitiesCaseSensitiveSourceField": false,
 "linkOpportunitiesCaseSensitiveMatchField": false,
 "linkOpportunitiesMultipleSourceMatches": true,
 "name": "Opportunity Contact Linkage Import",
 "fields": {
  "EmailAddress": "{{Opportunity.Contact.Field(C_EmailAddress)}}",
  "OpportunityID": "{{Opportunity.Id}}"
 },
 "identifierFieldName": "OpportunityID",
 "isSyncTriggeredOnImport": false,
 "dataRetentionDuration": "P7D",
 "uri": "/opportunities/contacts/imports/23",
 "createdBy": "API.User",
 "createdAt": "2018-02-15T15:03:43.3345307Z",
 "updatedBy": "API.User",
 "updatedAt": "2018-02-15T15:03:43.3345307Z"
}
```

2. Upload data to the definition.

```
POST /api/bulk/2.0/opportunities/contacts/imports/23/data
Content-Type: application/json
```

Request body

```
[
 {
```

```
  "OpportunityID": "1",
  "EmailAddress": "contact1@abccompany.com"
 },
 {
  "OpportunityID": "2",
  "EmailAddress": "contact2@zyxcompany.com"
 }
]
```

All of my contacts with the email address contact1@abccompany.com are being linked to the opportunity with OpportunityID #1. And all of my contacts with the email address contact2@zyxcompany.com are being linked to the opportunity with OpportunityID #2.

Response

```
204 No Content
```

3. Synchronize the data.

```
POST /api/bulk/2.0/syncs
Content-Type: application/json
```

Request body

```
{
  "syncedInstanceUri": "/opportunities/contacts/imports/23"
}
```

Response

```
{
  "syncedInstanceUri": "/opportunities/contacts/imports/23",
  "status": "pending",
  "createdAt": "2018-02-15T15:04:55.3345307Z",
```

```
  "createdBy": "API.User",
  "uri": "/syncs/13"
}
```

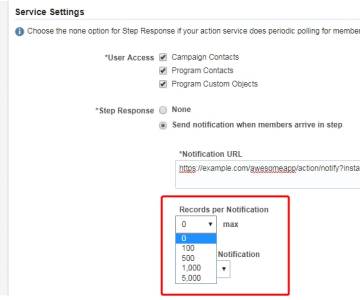
When the sync is successful, the response will resemble:

```
{
  "syncedInstanceUri": "/opportunities/contacts/imports/23",
  "syncStartedAt": "2018-02-15T15:04:55.3345307Z",
  "status": "success",
  "createdAt": "2018-02-15T15:04:55.3345307Z",
  "createdBy": "API.User",
  "uri": "/syncs/13"
}
```

Retrieving app records using the bulk API

You can get notified when contacts or custom objects arrive to your app within a campaign or program. Eloqua will send your app a notification call in the form of a HTTP POST request that includes data about the contacts or custom object records. This is accomplished by supplying a **Notification URL** during service registration for action and decision services.

You control the maximum amount of contacts or custom object records sent per notification call by setting the **Records per Notification** option during service registration.



When **Records per Notification** is greater than 0, the HTTP POST request sent to the app will include data for the contacts or custom objects within the `items` array.

See an example

```
POST https://example.com/awesomeapp/action/notify?instance=
{InstanceId}&asset={AssetId}&execution={ExecutionId}

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 2,
   "count" : 2,
   "hasMore" : false,
   "items" :
   [
     {
       "ContactID" : "1",
       "EmailAddress" : "fred@example.com",
       "field1" : "stuff",
       "field2" : "things",
       "field3" : "et cetera"
     },
     {
       "ContactID" : "2",
       "EmailAddress" : "john@example.com",
       "field1" : "more stuff",
       "field2" : "other things",
       "field3" : "and so on"
     }
   ]
}
```

This request can get large, especially if the maximum records sent per notification is 5,000 for example. It is possible instead for app developers to select the maximum amount of **Records per Notification** to 0 and retrieve records on their own.

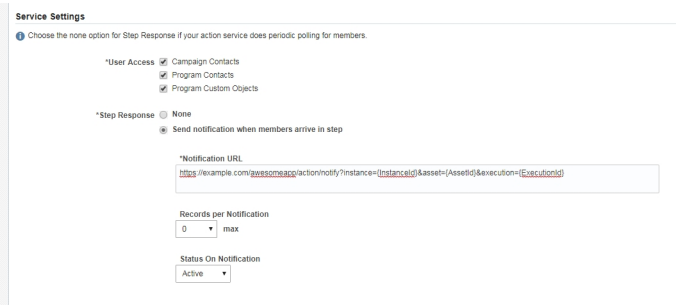
Setting records per notification to 0

If **Records per Notification** is set to 0, the notification call is sent with no data within the `items` property. The developer must then retrieve the records within the step using a separate bulk API export. This enables developers to control when and how quickly records are retrieved.

This tutorial will walkthrough the flow of having **Records per Notification** set to 0 and guiding developers through the flow of retrieving records within a step using a separate bulk API export.

Eloqua app service settings

Before we walkthrough how to respond to the notification call, note the Eloqua app service settings we will be using for the examples in this tutorial.



- Note the notification URL contains the 3 template parameters: `InstanceId` , `AssetId`, and `ExecutionId`.
- **Records per Notification** is set to **0**.

To edit your service settings, navigate to **Settings** > **AppCloud Developer** and edit your service's settings.

Notification call

When contacts or custom objects arrive in the step, Eloqua sends a HTTP POST request to the app's Notification URL.

```
POST https://example.com/awesomeapp/action/notify?instance=f82d50cd-86a9-4fca-b37e-4ec9a98b0339&asset=456&execution=12345

{
   "offset" : 0,
   "limit" : 1000,
   "totalResults" : 0,
   "count" : 0,
   "hasMore" : false,
   "items" : []
}
```

Developers must then create a bulk API export definition using the data from Eloqua's call to the Notification URL.

Using contacts as an example, let's walkthrough how to form our bulk API export definition to retrieve the records.

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/exports

{
   "name": "Awesome App Contact Export - f82d50cd-86a9-4fca-b37e-4ec9a98b0339 - 12345",
   "fields": {
      "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
     "filter": "STATUS('{{ActionInstance (f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}') = 'pending'"
}
```

Response


```
{
   "name": "Awesome App Contact Export - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
   "fields": {
      "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
   },
   "dataRetentionDuration": "PT12H",
   "uri": "/contacts/exports/55",
   "createdBy": "API.User",
   "createdAt": "2018-08-19T20:51:28.8201911Z",
   "updatedBy": "API.User",
   "updatedAt": "2018-08-19T20:51:28.8201911Z"
}
```


Next, create a sync using the export definition uri.

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/syncs

{
  "syncedInstanceUri" : "/contacts/exports/55",
  "callbackURL" : "https://www.example.com"
}
```

Response


```
{
   "syncedInstanceUri":"/contacts/exports/55",
   "callbackUrl":"http://www.example.com",
   "status":"pending",
   "createdAt":"2018-09-25T18:08:32.3485942Z",
   "createdBy":"API.User",
   "uri":"/syncs/66100"
}
```

The sync `status` will progress. Retrieve the sync to check the status of the sync.

Request

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/sync/66100
```

Response

```
{
  "syncedInstanceURI": "/contacts/exports/55",
  "syncStartedAt": "2014-01-01T13:59:07.1375620Z",
  "syncEndedAt": "2014-01-01T13:59:14.1375620Z",
  "status": "success",
  "createdAt": "2014-01-01T13:59:07.1375620Z",
  "createdBy": "DocsExample",
  "uri": "/syncs/6"
}
```

After the sync has completed and the status is `success`, the last step is to retrieve the sync data.

Request

```
GET https://secure.p03.eloqua.com/api/bulk/2.0/syncs/66100/data
```

Response

```
{
  "totalResults": 1,
  "limit": 0,
  "offset": 0,
  "count": 0,
  "hasMore": true,
  "items": [
    {
      "email": "john.snow@example.com"
```

```
  }
 ]
}
```

Using the data

Your app will likely want to use the records for a specific purpose, after this is done, you can import the data into Eloqua.

Importing the data into Eloqua

Start by creating a bulk import definition, setting the `status` to `complete` to import data. For this example, we will demonstrate using a contact import definition, where our service is an action service and its instance GUID is `f82d50cd-86a9-4fca-b37e-4ec9a98b0339`.

If there is no data to import, and you only need to update a record's status, you can update a record's status without performing an import by creating a contact sync action definition.

> ❑ **Warning**: When referencing service instances, you must transmit the GUID without dashes. The bulk API will error if you transmit the GUID with the dashes.

Create a contact import definition.

Request

```
POST https: //secure.p03.eloqua.com/api/bulk/2.0/contacts/imports
{
```

```
 "name": "Awesome App Contact Import - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
 "updateRule": "always",
 "fields": {
  "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "syncActions": [
  {
   "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
   "action": "setStatus",
   "status": "complete"
  }
 ],
 "identifierFieldName": "emailAddress"
}
```

Eloqua's response will be a 201 Created response that includes a `uri` parameter, which you can use to identify the import.

Response

```
{
 "name": "Awesome App Contact Import - f82d50cd-86a9-4fca-b37e-
4ec9a98b0339 - 12345",
 "updateRule": "always",
 "fields": {
  "emailAddress": "{{Contact.Field(C_EmailAddress)}}"
 },
 "identifierFieldName": "emailAddress",
 "syncActions": [
  {
   "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
   "action": "setStatus",
   "status": "complete"
  }
 ],
```

```
  "isSyncTriggeredOnImport": false,
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/contacts/imports/6",
  "createdBy": "API.User",
  "createdAt": "2018-03-06T13:59:00.6600046Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-03-06T13:59:00.6600046Z"
}
```

Send Eloqua the import data using the `uri`:

Request

```
POST https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/6/data
[
  {
    "emailAddress": "john.snow@example.com"
  }
]
```

Eloqua's response will be a 204 No Content indicating that the data upload was successful.

Uploading file data using cURL

If you want to upload data using the bulk API, you can use a cURL request to upload data from a .json or .csv file located on your machine. This differs from uploading data with JSON directly in the cURL request and is more suitable for larger amounts of data.

This tutorial will explain how to format your cURL request to upload data from a .json and .csv file on your machine. If you are not yet familiar with sending API requests using cURL, see the tutorial.

In this tutorial:

- Uploading data from a JSON file
- Uploading data from a CSV file

To upload data from a file:

Let's start by creating a contact import definition to be used to upload data.

```
POST /bulk/2.0/contacts/imports

{
  "name": "Contact File Import",
  "fields": {
    "Email": "{{Contact.Field(C_EmailAddress)}}",
    "FirstName": "{{Contact.Field(C_FirstName)}}"
  },
  "identifierFieldName": "Email",
  "isSyncTriggeredOnImport": true
}
```

Here's the response.

```
{
  "name": "Contact File Import",
  "fields": {
    "Email": "{{Contact.Field(C_EmailAddress)}}",
    "FirstName": "{{Contact.Field(C_FirstName)}}"
  },
  "identifierFieldName": "Email",
  "syncActions": [],
  "isSyncTriggeredOnImport": true,
  "dataRetentionDuration": "P7D",
  "kbUsed": 0,
  "isUpdatingMultipleMatchedRecords": false,
  "uri": "/contacts/imports/30",
  "createdBy": "API.User",
  "createdAt": "2018-09-18T20:15:11.5900000Z",
  "updatedBy": "API.User",
  "updatedAt": "2018-09-18T20:15:11.5900000Z"
```

```
}
```

Let's walkthrough how to upload data to this definition from a file.

Uploading data from a .json file

You'll need a .json file on your machine. Our .json file is named "apitest.json" and it contains the following:

```
[
 {
  "Email": "john.doe@oracle.com",
  "FirstName": "John"
 },
 {
  "Email": "billy.bishop@oracle.com",
  "FirstName": "Billy"
 }
]
```

Using bash for our command line tool, navigate to the directory where your .json file is located.

```
cd C:\files
```

Next, we're going to send a POST request using cURL to upload data to the import definition we created with the id = 30.

```
curl --user "APITest\API.User" --header "Content-Type: application/json" --request POST --data "@apitest.json"
https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/30/data
```

Let's take a closer look at what's happening in the request as it differs from uploading data with JSON directly in the cURL request.

- The `--data` option is used with the file name of our .json file `--data "@apitest.json"`
- The `Content-Type` header is set to `application/json`

Because we set `isSyncTriggeredOnImport` to `true` for our import definition, the data was synced for us. The last step is to check the status of the sync.

Uploading data from a .csv file

You'll need a .csv file on your machine. Our .csv file is named "apitest.csv" and it contains the following:

```
Email,FirstName
john.doe@oracle.com,John
billy.bishop@oracle.com,Billy
```

Using bash for our command line tool, navigate to the directory where your .csv file is located.

```
cd C:\files
```

Next, upload data to the import definition we created earlier. Note that our .csv file is named "apitest.csv" and the Content-Type header is set to "text/csv".

```
curl --user "APITest\API.User" --header "Content-Type: text/csv" --request POST --data-binary "@apitest.csv"
https://secure.p03.eloqua.com/api/bulk/2.0/contacts/imports/30/data
```

Let's take a closer look at what's happening in the request as it differs from uploading data with JSON directly in the curl request.

- The `--data-binary` option is used with the file name of our .csv file `--data-binary "@apitest.csv"`
- The `Content-Type` header is set to `text/csv`

Because we set `isSyncTriggeredOnImport` to `true` for our import definition, the data was synced for us. The last step is to check the status of the sync.

Bulk API Best Practices

Here are a list of best practices for using the Bulk API.

Reusing definitions

A Bulk API definition can be reused as long as there is not an active sync using that definition. If there is any chance a definition is still in an active state in a sync, a new definition should be used. Syncing the same definition when the sync is in progress in another sync could lead to syncs failing, or data not being imported or exported due to conflicts.

Retrieving data

To retrieve data, always use the /syncs/<id>/data endpoint and not the definition's data endpoint. There are several issues with the definition's data endpoint, and using this endpoint can cause performance issues and result in incomplete data.

Checking a sync's status

It's recommended to avoid polling to check the sync's status, and instead use the sync endpoint's `callbackUrl` parameter. When a sync completes, Eloqua will make a call to the URL specified in this parameter. Learn more about the sync endpoint.

Exporting activities

We recommend following these best practices when exporting activities. For an in-depth explanation of exporting activities, see this Code It post.

- **Use filters to ensure fewer than 5,000,000 records are exported per Bulk API sync**

  A common method of filtering is to use the Activity Date to export a finite time period of activity, such as one month (use larger or smaller time periods based on volume of activity generation).

- **Export one Activity Type at a time**

  Do not run syncs for all Activity Types simultaneously.

- **Make multiple export requests sequentially**

  Allow the sync to finish before executing the next sync.

Important:

- There is a maximum of 5 million records per Activity export sync. If necessary, you'll need to use a date/time filter to reduce the number of records exported to below 5 million. Activity volume can be checked using Insight reports. No filter is needed if there are fewer than 5 million records. Learn more about Bulk filtering.
- There is a maximum of 10 contact fields allowed in an activity export definition. The addition of contact fields to activity exports will add to export time.
- Contact fields cannot be used in the filter.
- Bouncebacks are recorded asynchronously, which means the Bulk API Activity Created Date (when the bounceback occurred), may not be available to export until hours or

days after the bounceback occurred. The upper date limit for a filter should be at least one hour before the time of Bulk API sync creation to ensure the majority of bouncebacks have been recorded.

Importing

We recommend following these best practices when importing data into Eloqua. For an in-depth explanation of importing, see this topic.

- Perform incremental import of data to avoid large imports, e.g. if there are no changes with a record since the last import, do not import that record.
- Schedule large imports during off-peak hours, so it does not hold up the queue. Off-peak hours can be confirmed by looking at Bulk API Syncs in the Marketing Operations Center.

Exporting contacts in a segment or shared filter

Executing Segments or Shared Filters is a heavy operation, so exporting contacts in a Segment or Shared Filter via the Bulk API should be done as infrequently as needed (e.g. once a day), and if exporting multiple Segments or Shared Filters, they should be performed sequentially.

Eloqua expression language

The Eloqua expression language (EEL) support for incorporating logic into decision making and filtering within Eloqua replaces the `ExportFilter` data transfer object that was present in Bulk 1.0. EEL simplifies and formalizes the syntax, providing a consistent way to interact with your data.

Operators

Comparison Operators

- **>**
- **>=**
- **<**
- **<=**
- **!=**
- **=**
- **~** (This is the SQL Server LIKE operator. see: Like (Transact-SQL) on the Microsoft Developer Network documentation for appropriate syntax.)

Logical operators

- `AND`
- `OR`
- `NOT`

Existence operators

EEL includes the `EXISTS` and `STATUS` operators as ways to filter based on a contacts' presence or absence in a container, or on the status of associated containers. Unlike the comparison and logical operators, `EXISTS` and `STATUS` can only act on certain entities.

> **Note**: You can only use one EXISTS and STATUS operator in a filter.

EXISTS

`EXISTS` is analogous to an `IN` operator. It determines if the object or entity exists within a container. The valid entities are: contact filters, contact segments, contact lists, and account Lists, as in the following examples:

```
EXISTS('{{ContactFilter[<id>]}}')
EXISTS('{{ContactSegment[<id>]}}')
EXISTS('{{ContactList[<id>]}}')
EXISTS('{{AccountList[<id>]}}')
```

The `<id>` being the entity's `id` (e.g. 12).

> **Note**: `EXISTS` operators can only be used on exports of the same entity:
>
> - The ContactList, ContactFilter, or ContactSegment `EXISTS` operators can only be used in filters for contact exports.
> - The AccountList `EXISTS` operator can only be used in filters for account exports.

STATUS

`STATUS` describes the current status of a cloud connector, or AppCloud action, content, or decision service. `STATUS` does not currently support accessing email groups, campaigns, and so on.

For cloud connectors, use the following format:

```
STATUS('CloudConnectorStatus(<instanceid>)') = '<status>'
```

Where the `<instanceid>` is the instance's ID, and `<status>` is either active or pending.

For AppCloud actions and decisions, the following format is correct:

```
STATUS('{{DecisionInstance(<instanceId>).Execution[<executionId>]}}') = '<status>'
STATUS('{{ActionInstance(<instanceId>).Execution[<executionId>]}}') = '<status>'
```

For AppCloud feeders the following format is correct:

```
STATUS('{{FeederInstance(<instanceId>)}}') = '<status>'
```

The `<instanceId>` being the GUID for the specific instance of the AppCloud service being used, and the `<executionId>` being the integer identifying a unique AppCloud service instance's execution. The `<status>` can be active, pending, complete, or errored.

The `STATUS` can also be `invalid` for AppCloud actions and decisions.

Examples

> **Important:** Wrap all operands in single quotes.

Comparison

Select contacts whose `CreatedAt` field is greater that 2018-12-31:

```
"filter" : "'{{Contact.CreatedAt}}' > '2018-12-31'"
```

Select contacts whose Account's Company Name is not Eloqua:

```
"filter" : "'{{Contact.Account.Field(M_CompanyName)}}' != 'Eloqua'"
```

Select contact with email address "test.a'pie@test.test":

```
"filter": "'{{Contact.Field(C_EmailAddress)}}'='test.a\\'pie@test.test'"
```

Logical

> **Important:**
>
> Supported filter formats with logical operators:
>
> - (A OR B) AND (C OR D)
> - A AND NOT B AND (C OR D)
> - A AND B AND (C OR D)
> - A AND (B OR C)
>
> Activity exports only support the A AND B AND C filter format.

Select contacts whose `CreatedAt` field is greater that 2018-12-31 and whose account's company name is not Eloqua:

```
"filter" : "'{{Contact.CreatedAt}}' > '2018-12-31' AND '{{Contact.Account.Field(M_
CompanyName)}}' != 'Eloqua'"
```

Select contacts whose `CreatedAt` field is greater than 2018-12-31 and whose account's company name is not Eloqua and whose `C_Country` field is Canada or United States:

```
"filter": "'{{Contact.CreatedAt}}' > '2018-12-31' AND '{{Contact.Account.Field(M_CompanyName)}}' != 'Eloqua' AND ('{{Contact.Field(C_Country)}}' = 'CA' OR '{{Contact.Field(C_Country)}}' = 'US')"
```

Existence

`EXISTS` refers specifically to a container. The following filters contacts based on their presence in the ContactList with `id` 123:

```
"filter" : "EXISTS('{{ContactList[123]}}')"
```

`STATUS`. The following filter selects records for which the AppCloud action service instance ID is f82d50cd86a94fcab37e4ec9a98b0339, with an execution ID of 12345 and a status is pending:

```
"filter" : "STATUS('{{ActionInstance(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}') = 'pending'"
```

> **Note**: Filters have a 1000 character limit. Contact fields cannot be used in the filter for activity exports.

Grammar

```
grammar Ebl;

options {
  language = CSharp3;
  output = AST;
  backtrack = true;
}
```

```
@modifier{public}
@parser::namespace { Eloqua.Expression.Version1.Grammar }
@lexer::namespace { Eloqua.Expression.Version1.Grammar }

public expression
  : WS!? orExpression WS!? EOF
  ;

orExpression
  : andExpression (WS!? OR^ WS!? andExpression)*
  ;

andExpression
  : notExpression (WS!? AND^ WS!? notExpression)*
  ;

notExpression
  : (NOT WS!?)? atom
  ;

subExpression
  : OPENPAREN! orExpression WS!? CLOSEPAREN!
  ;

atom
  : comparison
  | subExpression
  ;

comparison
  : STRING WS!? GREATERTHAN WS!? STRING
  | STRING WS!? GREATERTHANOREQUAL WS!? STRING
  | STRING WS!? LESSTHAN WS!? STRING
  | STRING WS!? LESSTHANOREQUAL WS!? STRING
  | STRING WS!? NOTEQUAL WS!? STRING
  | STRING WS!? EQUAL WS!? STRING
  | STRING WS!? LIKE WS!? STRING
  | EXISTS WS!? OPENPAREN! WS!? STRING WS!? CLOSEPAREN!
  | STATUS WS!? OPENPAREN! WS!? STRING WS!? CLOSEPAREN! WS!? EQUAL! WS!?
 STRING
  ;
```

```
// Lexer tokens

// Comparison Operators
GREATERTHAN      : '>' ;
GREATERTHANOREQUAL : '>=' ;
LESSTHAN         : '<' ;
LESSTHANOREQUAL   : '<=' ;
NOTEQUAL         : '!=' ;
EQUAL            : '=' ;
LIKE             : '~' ;

// Existence Operators
EXISTS           : E X I S T S ;
STATUS           : S T A T U S ;

// Binary
AND              : A N D;
OR               : O R;

// Unary
NOT              : N O T;

// Grouping
OPENPAREN        : '(' ;
CLOSEPAREN       : ')' ;

// Lexer rules
STRING   : QUOTE ( ESC_SEQ | ~('\\'|'\'') )* QUOTE;
WS       : (' '|'\t'|'\n'|'\r')+ ;

fragment
ESC_SEQ
   :  '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')
   |  '\\' 'u' HEX_DIGIT HEX_DIGIT HEX_DIGIT HEX_DIGIT
   ;

fragment QUOTE    : '\'' ;
fragment HEX_DIGIT : ('0'..'9'|'a'..'f'|'A'..'F') ;

fragment A:('a'|'A');
```

```
fragment  B:('b'|'B');
fragment  C:('c'|'C');
fragment  D:('d'|'D');
fragment  E:('e'|'E');
fragment  F:('f'|'F');
fragment  G:('g'|'G');
fragment  H:('h'|'H');
fragment  I:('i'|'I');
fragment  J:('j'|'J');
fragment  K:('k'|'K');
fragment  L:('l'|'L');
fragment  M:('m'|'M');
fragment  N:('n'|'N');
fragment  O:('o'|'O');
fragment  P:('p'|'P');
fragment  Q:('q'|'Q');
fragment  R:('r'|'R');
fragment  S:('s'|'S');
fragment  T:('t'|'T');
fragment  U:('u'|'U');
fragment  V:('v'|'V');
fragment  W:('w'|'W');
fragment  X:('x'|'X');
fragment  Y:('y'|'Y');
fragment  Z:('z'|'Z');
```

Eloqua markup language version 2

> **Important**: This document discusses Eloqua markup language v2. Learn more about Eloqua markup language v3.

Eloqua markup language (EML) is a text-based language used to access and provide a consistent view of Eloqua data.

EML uses text to identify fields and data relationships for data-centric entities. This way, you can easily describe and access the field values, entity attributes and relations that are important to you, using intuitive language.

Terminology

- **Contact**: A contact in the Eloqua system
- **Account**: An account or company in the Eloqua system
- **CustomObject**: A custom object in the Eloqua system, with its own set of field definitions
- **Field**: An attribute of a Contact, Account or Custom Object that has an identifier and an internal name

Access patterns

All statements enclosed in `{{` and `}}` will be resolved if they can be parsed. If there is no available data an empty string will be inserted.

For all entities that can be indexed, indexing is available by ID or system name. The pattern is to use `(` and `)` to delimit a named index, and `[` and `]` to delimit an ID index. Note that because the wrong brackets are used, the following is **invalid** and cannot be indexed: `{{Contact.Field(1000001)}} {{Contact.Field[C_EmailAddress]}}`

Related assets are addressed through a "`.`" notation. For example, to get the data for the account name field of the account related to a given contact, you could use the following statement (assuming there is an account field with the internal name **M_CompanyName**): `{{Contact.Account.Field(M_CompanyName)}}`.

Available entities

EML can access core assets: *Contact*, *Account* and *Custom Object* data.

Note: Access to additional core assets (Event, Campaign response, and Opportunity) were added in Eloqua markup language v3.

Assets related to core assets are available as long as the relationship resolves to a single entity and not many.

Example: Account information can be accessed through a contact since there is at most a single account related to any given contact, it resolves to a single entity. However, it doesn't work the other way around: contact information cannot likely be accessed through an account, as there is usually more than a single associated contact, i.e. `{{Contact.Account.Field(M_CompanyName)}}` is valid, but `{{Account.Contact.Field(C_FirstName)}}` is not.

Special fields are available for contact records:

- Global Subscription information is addressable as: `{{Contact.Email.IsSubscribed}}`
- Email Bounceback status is addressable as: `{{Contact.Email.IsBounced}}`
- Email Format preference is addressable as: `{{Contact.Email.Format}}`

Note: When using `{{Contact.Email.IsSubscribed}}` or `{{Contact.Email.IsBounced}}` in an export filter or in an import use “0” for False and “1” for True.

Resolution

EML parts are evaluated differently. Below are example resolutions for different EML types.

Field values

Resolves to the appropriate database value.

- **Example**: The definition `{{Contact.Field(C_FirstName)}}` resolves to `Haruki`.
- **Example**: The definition `{{Contact.Id}}` Resolves to `123456` (the internal contact identifier).

Multiple entities can be related to a single value.

- **Example**: The definition `{{Contact.Account.Field(M_CompanyName)}}` resolves to `Eloqua`.

Field values can also be combined.

- **Example**: The definition "Mr. `{{Contact.Field(C_FirstName)}}` `{{Contact.Field(C_LastName)}}`" Resolves to "`Mr. Haruki Murakami`".

Example: There is a contact field in the system with the below properties. You can retrieve this information by calling a GET request to the "`/assets/contact/fields?depth=complete`" endpoint.


```
"type": "ContactField",
"id": "100001",
"createdAt": "-2208970800",
"depth": "complete",
"name": "Email Address",
"updatedAt": "-2208970800",
"dataType": "text",
"displayType": "text",
"internalName": "C_EmailAddress",
"isReadOnly": "false",
"isRequired": "false",
"isStandard": "true",
"isPopulatedInOutlookPlugin": "false",
"updateType": "always"
```


The following contact data is retrieved by calling a GET request to the "`/data/contact/1`" endpoint:


```
"type":"Contact",
"currentStatus":"Awaiting action",
"id":"1",
"createdAt":"1403034086",
"depth":"complete",
"name":"haruki.murakami@oracle.com",
"updatedAt":"1410193024",
```

```
    "emailAddress":"haruki.murakami@oracle.com",
    "emailFormatPreference":"unspecified",
    "fieldValues":[ ... ],
    "firstName":"haruki",
    "isBounceback":"false",
    "isSubscribed":"true",
    "lastName":"murakami",
    "subscriptionDate":"1403034086"
```

Given the above information, the following EML statements: `{{Contact.Field[100001]}}` and `{{Contact.Field(C_EmailAddress)}}` will resolve to the value of the specified field, which in both cases is the contact's email address: "`haruki.murakami@oracle.com`". `{{Contact.Field(C_EmailAddress)}}` references the asset's email address field by its `internalName` attribute, and `{{Contact.Field[100001]}}` references the asset's email address by its field id.

Grammar

```
grammar Eml;

options {
   language = CSharp3;
   output = AST;
   backtrack = true;
}

tokens {
   CUSTOMOBJECT_NAME;
   CUSTOMOBJECT_INDEX;
   FIELD_NAME;
```

```
    FIELD_INDEX;
    ACTIVITY_TYPE_ATTRIBUTE;
    ASSET_NAME_ATTRIBUTE;
    ASSET_ID_ATTRIBUTE;
    CAMPAIGN_ID_ATTRIBUTE;
}

public dynamicEntity
    : DYNAMICSTART! WS!? ( (relationshipToContact WS!? DOT! WS!? contactAttribute)
        | (relationshipToAccount WS!? DOT! WS!? accountAttribute)
        | (relationshipToCustomObject WS!? DOT! WS!? customObjectAttribute)
        | (relationshipToActivity WS!? DOT! WS!? activityAttribute)
    ) WS!? DYNAMICEND!
    ;

relationshipToContact
    : CONTACT
    | customObject WS!? DOT! WS!? CONTACT
    | ACTIVITY WS!? DOT! WS!? CONTACT
    ;

relationshipToAccount
    : ACCOUNT
    | CONTACT WS!? DOT! WS!? ACCOUNT
    | customObject WS!? DOT! WS!? ACCOUNT
    ;

relationshipToCustomObject
    : customObject
    ;

relationshipToActivity
    : ACTIVITY
  ;

contactAttribute
    : field
    | ID_ATTRIBUTE
    | EMAIL! WS!? DOT! WS!? (
        | EMAIL_ISSUBSCRIBED_ATTRIBUTE
```

```
      | EMAIL_ISBOUNCED_ATTRIBUTE
      | EMAIL_FORMAT_ATTRIBUTE
   );

accountAttribute
   : field
   | ID_ATTRIBUTE
   ;

customObjectAttribute
   : field
   | ID_ATTRIBUTE
   ;

activityAttribute
   : TYPE_ATTRIBUTE -> ACTIVITY_TYPE_ATTRIBUTE
  | CREATEDAT_ATTRIBUTE
  | CAMPAIGN WS? DOT WS? ID_ATTRIBUTE -> CAMPAIGN_ID_ATTRIBUTE
   | ASSET WS? DOT WS? (
      | NAME_ATTRIBUTE -> ASSET_NAME_ATTRIBUTE
      | ID_ATTRIBUTE -> ASSET_ID_ATTRIBUTE
   );

field
   : FIELD WS? (
        NAMESTART WS? NAME WS? NAMEEND -> FIELD_NAME NAME
      | INDEXSTART WS? INT WS? INDEXEND -> FIELD_INDEX INT
   );

customObject
   : CUSTOMOBJECT WS? (
        NAMESTART WS? NAME WS? NAMEEND -> CUSTOMOBJECT_NAME NAME
      | INDEXSTART WS? INT WS? INDEXEND -> CUSTOMOBJECT_INDEX INT
   );

// Lexer tokens
DYNAMICSTART : '{{' ;
DYNAMICEND   : '}}' ;
INDEXSTART   : '['  ;
INDEXEND     : ']'  ;
```

```
NAMESTART   : '('  ;
NAMEEND     : ')'  ;
DOT         : '.'  ;

CONTACT                 : 'Contact';
ACCOUNT                 : 'Account';
CUSTOMOBJECT                : 'CustomObject';
ACTIVITY                : 'Activity';
FIELD                   : 'Field';
EMAIL                   : 'Email';
ASSET                   : 'Asset';
CAMPAIGN                : 'Campaign';
ID_ATTRIBUTE            : 'Id';
EMAIL_ISSUBSCRIBED_ATTRIBUTE : 'IsSubscribed';
EMAIL_ISBOUNCED_ATTRIBUTE    : 'IsBounced';
EMAIL_FORMAT_ATTRIBUTE      : 'Format';
TYPE_ATTRIBUTE          : 'Type';
NAME_ATTRIBUTE          : 'Name';
CREATEDAT_ATTRIBUTE         : 'CreatedAt';

NAME : ('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_'|' ')* ;
INT  : ('0'..'9')+ ;
WS   : (' '|'\t'|'\n'|'\r')+ ;
```

Eloqua markup language version 3

> **Important**: This topic discusses Eloqua markup language v3. Learn more about Eloqua markup language v2.

Eloqua markup language (EML) is a text-based language used to access and provide a consistent view of Eloqua data.

EML uses text to identify fields and data relationships for data-centric entities. This way, you can easily describe and access the field values, entity attributes and relations that are important to you, using intuitive language.

Available entities

EML allows you to access Contact, Account, Custom object, Event, Campaign response, and Opportunity data within your Eloqua database. You can also access related assets, providing the relationship resolves to a single entity.

Access patterns

The general access format is `{{EntityName.InfoYouWant}})`, where `InfoYouWant` is a field name or attribute. The syntax varies slightly between those two types of data: for *named fields*, the syntax is `{{EntityName.Field(FieldName)}}`, while for *attributes*, the syntax is `{{EntityName.AttributeName}}`.

EML also uses dot notation to connect related assets, in the form `{{EntityName.RelatedAssetName.InfoYouWant}}`.

There are some syntax requirements to bear in mind:

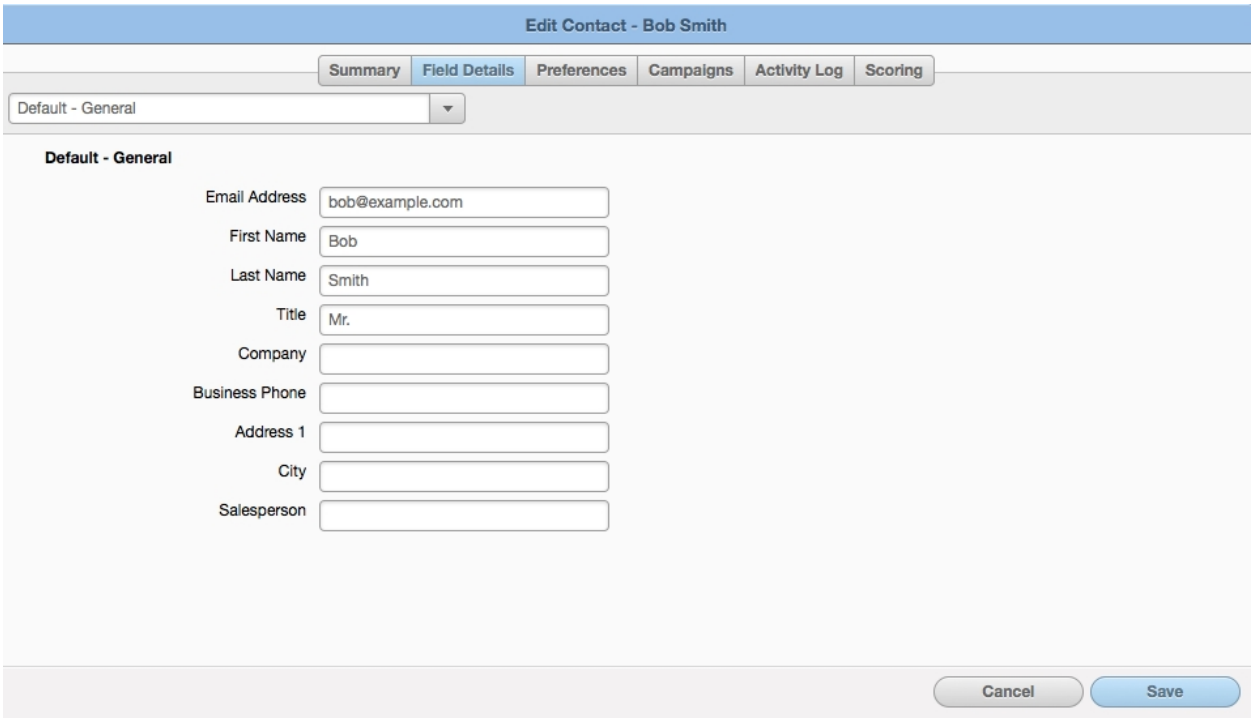
- The system resolves all statements enclosed in `{{` and `}}` if it can parse them. You must put statements that require resolution between double braces.
- To access *attributes*, use brackets `[ ]` to delimit the requested attribute. To access *fields*, use parentheses `( )` to delimit the named field.
- When specifying a field or asset using elm, you must refer to the *internal name*, rather than the *display name*. the display name is the name that appears in the eloqua ui, while the internal name is the identifier used by the system. the contact entity's email address field,

for example, has the display name: "email address" and the internal name "c_emailaddress". see Field names for more information.

You can only access a related asset that resolves to a single record. For instance, since a Contact is associated with at most one Account, you can retrieve a Contact's related Account information. Conversely, since Accounts are linked to many Contacts, you cannot retrieve related Contact information for an Account.

Syntax examples

These examples use a fictional contact called Bob Smith throughout. Bob's contact information resembles the following:



Field values

To access field values, the format is `{{Type.Field(fieldName)}}`. For instance, to access the `C_FirstName` field in a Contact, the markup would resemble:

```
{{Contact.Field(C_FirstName)}}
```

For contact Bob Smith, the definition resolves to *Bob*.

Entity attributes

You can also access an entity's attributes. The definition is extremely similar to the markup for accessing entity fields. To access a contact's `Id`, the internal contact identifier that Eloqua generates and stores when you add a new contact, you would input:

```
{{Contact.Id}}
```

For Bob Smith, whose internal contact identifier is 123456, `{{Contact.Id}}` resolves to *123456*.

Related entities

- To access a Contact's linked Account information, in this case, `M_CompanyName`, the definition is:

  ```
  {{Contact.Account.Field(M_CompanyName)}}
  ```

  For Bob Smith, who is a contact of Eloqua, this resolves to *Eloqua*.

- To access a Contact's field via activity export definition, in this case, `C_EmailAddress`, the definition is:

  ```
  {{Activity.Contact.Field(C_EmailAddress)}}
  ```

  For Bob Smith, whose email address is bob@example.com, this resolves to *bob@example.com*.

Changes in version 3

Eloqua markup language v3 builds on the previous version of EML to simplify and streamline the ways you describe and access data within Eloqua. EML v3 adds support for:

- Accessing visitor information for activities coming in from the web. For example, when someone opens an email sent through Eloqua.
- Discovering lead scoring models and fields.
- Including lead scoring fields in a contact export definition, and filtering on lead scoring field values.
- Discovering events and event fields.
- Exporting event registrants.
- Importing event registrants.
- Including contact fields in an activity export definition.
- Discovering campaign response fields.
- Exporting and importing campaign responses.
- Discovering opportunity fields.
- Importing opportunities.
- Including campaign fields in an activity export definition.
- Including user fields in an activity export definition.
- Discovering campaign fields.
- Including campaign fields in a campaign response export definition.

Grammar

```
grammar Eml;

options {
   language = CSharp3;
   output = AST;
   backtrack = true;
}

tokens {
   CUSTOMOBJECT_NAME;
   CUSTOMOBJECT_INDEX;
   FIELD_NAME;
   FIELD_INDEX;
   ACTIVITY_TYPE_ATTRIBUTE;
   ASSET_TYPE_ATTRIBUTE;
   ASSET_NAME_ATTRIBUTE;
   ASSET_ID_ATTRIBUTE;
   CAMPAIGN_ID_ATTRIBUTE;
   VISITOR_ID_ATTRIBUTE;
   VISITOR_EXTERNAL_ID_ATTRIBUTE;
   USER_ID_ATTRIBUTE;
}

@modifier{public}
@parser::namespace { Eloqua.Markup.Version3.Grammar }
@lexer::namespace { Eloqua.Markup.Version3.Grammar }

public dynamicEntity
   : DYNAMICSTART! (dataEntity | assetEntity) DYNAMICEND!
   ;

assetEntity
      : contactContainerAsset
      | emailGroupAsset
      | accountContainerAsset
      | cloudInstanceAsset
      | emailAddressAsset
      ;
```

```
contactContainerAsset
      : CONTACTLIST INDEXED_ID
      | CONTACTFILTER INDEXED_ID
      | CONTACTSEGMENT INDEXED_ID
      ;

emailAddressAsset
      : GLOBALSUBSCRIBE
      ;

emailGroupAsset
      : EMAILGROUP INDEXED_ID
      | EMAIL INDEXED_ID DOT! GROUP
      ;

accountContainerAsset
      : ACCOUNTLIST INDEXED_ID
      ;

cloudInstanceAsset
      : CONTENTINSTANCE NAMED_ID (DOT! execution)?
      | ACTIONINSTANCE NAMED_ID (DOT! execution)?
      | DECISIONINSTANCE NAMED_ID (DOT! execution)?
      | FEEDERINSTANCE NAMED_ID
      ;

dataEntity
      : relationshipToContact DOT! contactAttribute
    | relationshipToContact DOT! LEADSCORE DOT leadScoreModel DOT!
leadScoreModelAttribute
    | relationshipToAccount DOT! accountAttribute
    | relationshipToCustomObject DOT! customObjectAttribute
    | relationshipToEvent DOT! eventAttribute
    | relationshipToActivity DOT! activityAttribute
    | relationshipToEmailAddress DOT! emailAddressAttribute
    | relationshipToCampaignResponse DOT! campaignResponseAttribute
    | relationshipToCampaign DOT! campaignAttribute
    | relationshipToUser DOT! userAttribute
    | opportunity DOT! opportunityAttribute
 ;
```

```
relationshipToContact
   : CONTACT
   | customObject DOT! CONTACT
   | event DOT! CONTACT
   | ACTIVITY DOT! CONTACT
   | CAMPAIGNRESPONSE DOT! CONTACT
   | opportunity DOT! CONTACT
   ;

relationshipToAccount
   : ACCOUNT
   | CONTACT DOT! ACCOUNT
   | customObject DOT! ACCOUNT
   | event DOT! ACCOUNT
   | opportunity DOT! ACCOUNT
   ;

relationshipToCustomObject
   : customObject
   ;

 relationshipToEvent
   : event
   ;

relationshipToActivity
   : ACTIVITY
     ;

relationshipToEmailAddress
     :     EMAILADDRESS
     ;

relationshipToCampaignResponse
   : CAMPAIGNRESPONSE
   ;

relationshipToCampaign
   : CAMPAIGNRESPONSE DOT! CAMPAIGN
```

```
    | ACTIVITY DOT! CAMPAIGN
    ;

relationshipToUser
      :      ACTIVITY DOT! USER
      ;

contactAttribute
    : field
    | ID_ATTRIBUTE
    | CREATEDAT_ATTRIBUTE
    | UPDATEDAT_ATTRIBUTE
    | EMAIL! DOT! (
        | EMAIL_ISSUBSCRIBED_ATTRIBUTE
        | EMAIL_ISBOUNCED_ATTRIBUTE
        | EMAIL_FORMAT_ATTRIBUTE
    )
    ;

opportunityAttribute
      : field
      | ID_ATTRIBUTE
      | CREATEDAT_ATTRIBUTE
      | UPDATEDAT_ATTRIBUTE
      ;

accountAttribute
    : field
    | ID_ATTRIBUTE
    | CREATEDAT_ATTRIBUTE
    | UPDATEDAT_ATTRIBUTE
    ;

customObjectAttribute
    : field
    | ID_ATTRIBUTE
    | EXTERNAL_ID_ATTRIBUTE
    | CREATEDAT_ATTRIBUTE
    | UPDATEDAT_ATTRIBUTE
    ;
```

```
eventAttribute
   : field
   | ID_ATTRIBUTE
   | EXTERNAL_ID_ATTRIBUTE
   | CREATEDAT_ATTRIBUTE
   | UPDATEDAT_ATTRIBUTE
   ;

leadScoreModelAttribute
   : RATING
   | PROFILESCORE
   | ENGAGEMENTSCORE
   ;

activityAttribute
   : field
   | ID_ATTRIBUTE
   | EXTERNAL_ID_ATTRIBUTE
      | TYPE_ATTRIBUTE -> ACTIVITY_TYPE_ATTRIBUTE
      | CREATEDAT_ATTRIBUTE
      | CAMPAIGN DOT ID_ATTRIBUTE -> CAMPAIGN_ID_ATTRIBUTE
      | VISITOR DOT (
         | ID_ATTRIBUTE -> VISITOR_ID_ATTRIBUTE
            | EXTERNAL_ID_ATTRIBUTE -> VISITOR_EXTERNAL_ID_ATTRIBUTE
      )
      | USER DOT ID_ATTRIBUTE -> USER_ID_ATTRIBUTE
   | ASSET DOT (
      | TYPE_ATTRIBUTE -> ASSET_TYPE_ATTRIBUTE
      | NAME_ATTRIBUTE -> ASSET_NAME_ATTRIBUTE
      | ID_ATTRIBUTE -> ASSET_ID_ATTRIBUTE
   );

emailAddressAttribute
   : field
   | ID_ATTRIBUTE
   | CREATEDAT_ATTRIBUTE
   | UPDATEDAT_ATTRIBUTE
   ;
```

```
campaignResponseAttribute
   : field
   | ID_ATTRIBUTE
   | CREATEDAT_ATTRIBUTE
   ;

campaignAttribute
   : field
   | ID_ATTRIBUTE
   ;

userAttribute
   : field
   | ID_ATTRIBUTE
   ;

execution
   : EXECUTION INDEXED_ID
     ;

field
   : FIELD (
            NAMED_ID -> FIELD_NAME NAMED_ID
           | INDEXED_ID -> FIELD_INDEX INDEXED_ID
     );

customObject
   : CUSTOMOBJECT (
       NAMED_ID -> CUSTOMOBJECT_NAME NAMED_ID
      | INDEXED_ID -> CUSTOMOBJECT_INDEX INDEXED_ID
   );

event
   : EVENT INDEXED_ID
   ;

leadScoreModel
   : LEADSCOREMODEL INDEXED_ID
   ;
```

```
opportunity
   : OPPORTUNITY
   ;

// Lexer tokens
DYNAMICSTART : '{{' ;
DYNAMICEND   : '}}' ;
DOT          : '.'  ;

// roots
CONTACT                 : 'Contact';
ACCOUNT                 : 'Account';
CUSTOMOBJECT              : 'CustomObject';
EVENT                 : 'Event';
OPPORTUNITY             : 'Opportunity';
ACTIVITY                : 'Activity';
CONTACTLIST             : 'ContactList';
CONTACTFILTER             : 'ContactFilter';
CONTACTSEGMENT            : 'ContactSegment';
ACCOUNTLIST             : 'AccountList';
EMAIL                 : 'Email';
EMAILGROUP              : 'EmailGroup';
CONTENTINSTANCE           : 'ContentInstance';
ACTIONINSTANCE            : 'ActionInstance';
DECISIONINSTANCE          : 'DecisionInstance';
FEEDERINSTANCE            : 'FeederInstance';
GLOBALSUBSCRIBE           : 'GlobalSubscribe';
EMAILADDRESS              : 'EmailAddress';
LEADSCORE               : 'LeadScore';
LEADSCOREMODEL            : 'Model';
CAMPAIGNRESPONSE          : 'CampaignResponse';

// attributes
GROUP                   : 'Group';
CAMPAIGN                : 'Campaign';
FIELD                 : 'Field';
ASSET                   : 'Asset';
VISITOR                 : 'Visitor';
ID_ATTRIBUTE            : 'Id';
EXTERNAL_ID_ATTRIBUTE     : 'ExternalId';
```

```
EMAIL_ISSUBSCRIBED_ATTRIBUTE : 'IsSubscribed';
EMAIL_ISBOUNCED_ATTRIBUTE    : 'IsBounced';
EMAIL_FORMAT_ATTRIBUTE       : 'Format';
TYPE_ATTRIBUTE               : 'Type';
NAME_ATTRIBUTE               : 'Name';
CREATEDAT_ATTRIBUTE          : 'CreatedAt';
UPDATEDAT_ATTRIBUTE          : 'UpdatedAt';
EXECUTION                    : 'Execution';
RATING                       : 'Rating';
PROFILESCORE                 : 'ProfileScore';
ENGAGEMENTSCORE              : 'EngagementScore';
USER                         : 'User';


NAMED_ID : '(' ('a'..'z'|'A'..'Z'|'0'..'9'|'_'|' ')+ ')';
INDEXED_ID : '[' ('1'..'9') ('0'..'9')* ']';
```

Export characteristics

Export characteristics are properties you use to control and describe an export with the bulk API.

The following characteristics are **required**:

- `name` (**string**): name of this import (maximum 100 characters).

The following characteristics are **optional**:

- `dataRetentionDuration` (**duration**): the length of time exported data should remain in the staging area. Bulk API 2.0 uses the ISO-8601 standard for specifying all durations. Valid values are anything from PT1H (1 hour) to P14D (2 weeks). This setting will default to PT12H (12 hours) if not explicitly set during export definition creation.
- `autoDeleteDuration` (**duration**): the length of time before the import or export definition will be automatically deleted. Bulk API 2.0 uses the ISO-8601 standard for specifying all durations. The minimum duration is PT1H (1 hour), there is no maximum. If the property is not set, automatic deletion will not occur. This is typically used for *one-time-use* import and

export definitions.

> **Note**: The export definition should not be deleted until after data is retrieved. If the export is of large volume, we recommend setting the `autoDeleteDuration` to a minimum of 1 day. This parameter applies to definitions only and does not affect synced data.

Import characteristics

Import characteristics are properties you use to control and describe an import with the bulk API. They are **optional**.

- `name` (**string**): name of this import (maximum 100 characters).
- `isSyncTriggeredOnImport` (Boolean): Whether a sync is automatically created by the system import. By default, `isSyncTriggeredOnImport` is `true`. If you set it to `false`, data that you push into the system will not be processed until you explicitly create a sync step for this import.
- `updateRule`: Indicates whether to update values in the Eloqua database from values imported with this import definition. The available options are:
  - `always`: Always update
  - `ifNewIsNotNull`: Update if the new value is not blank
  - `ifExistingIsNull`: Update if the existing value is not blank
  - `useFieldRule`: Use the rule defined at the field level

- `autoDeleteDuration` (duration): The length of time before the import or export definition will be automatically deleted. Bulk API 2.0 uses the ISO-8601 standard for specifying all durations. The minimum duration is PT1H (1 hour), there is no maximum. If the property is not set, automatic deletion will not occur. This is typically used for *one-time-use* export definitions. Import definitions, export definitions and synced data strictly honor this setting.

- `isUpdatingMultipleMatchedRecords` (Boolean): When set to `true`, if Eloqua finds multiple matching records for the identifier field, Eloqua will update all of the fields that match. If set to `false`, Eloqua updates based on its internal algorithm. For contact imports, the `isUpdatingMultipleMatchedRecords` property must be set to `false` in order to update `Contact.Field(C_EmailAddress)`.

- `identifierFieldName` (string): The field which will be used to identify the entity. Must be a string value, at least 1 character and at most 100 characters long. The following field types are not supported:

    - Large text
    - Date
    - Boolean

- `dataRetentionDuration` (duration): The length of time that unsynchronized data from this import should remain in the staging area. Bulk API 2.0 uses the ISO-8601 standard for specifying all durations. Valid values are anything from PT1H (1 hour) to P2W (2 weeks). This setting will default to P7D (7 days) if not explicitly set during export definition creation. Import definitions, export definitions and synced data strictly honor this setting.

Import updateRule parameter

Import definitions include an `updateRule` parameter, which specifies how Eloqua should handle updating records when you import a new value for an existing field. For example, if contact Sally Jones' email address in Eloqua is

`sally.jones9@example.com`, and you import a new email address for Sally Jones, the `updateRule` determines whether Eloqua should retain the existing email address or replace it with the new one.

The following rule types are available:

- `always`: Always update.
- `ifNewIsNotNull`: Update if the new value is not blank.
- `ifExistingIsNull`: Update if the existing value is blank.
- `useFieldRule`: Use the rule defined at the field level.

If you do not specify an `updateRule`, the rule type defaults to `always`.

Sync actions

In this topic:

Sync actions are parameters that you declare in an import or export definition that specify additional actions Eloqua should take when you import or export data.

The following are supported sync actions:

- Add contacts to contact list
- Remove contacts from contact list
- Subscribe contacts to email group
- Unsubscribe contacts from email group
- Contacts global subscribe
- Contacts global unsubscribe

- AppCloud decision response
- AppCloud action response
- AppCloud feeder response
- AppCloud decision response (program canvas only)
- AppCloud action response (program canvas only)
- AppCloud feeder response (program canvas only)
- Add accounts to account list
- Remove accounts from account list
- Email addresses global unsubscribe
- Email addresses set bounceback

Sync action reference for contacts

Add to contact list

Action	Destination	Status
add	`{{ContactList[<id>]}}`	n/a

Example:

```
{
  "action": "add",
  "destination": "{{ContactList[12345]}}"
}
```

Remove from contact list

Action	Destination	Status
remove	`{{ContactList[<id>]}}`	n/a

Example:

```
{
  "action": "remove",
  "destination": "{{ContactList[12345]}}"
}
```

Subscribe to email group

Action	Destination	Status
setStatus	`{{EmailGroup[<id>]}}`	`subscribed`

Example:

```
{
  "action": "setStatus",
  "destination": "{{EmailGroup[12345]}}",
  "status": "subscribed"
}
```

Unsubscribe from email group:

Action	Destination	Status
setStatus	`{{EmailGroup[<id>]}}`	`unsubscribed`

Example:

```
{
  "action": "setStatus",
  "destination": "{{EmailGroup[12345]}}",
  "status": "unsubscribed"
}
```

Global subscribe (to all email groups)

Action	Destination	Status
setStatus	`{{GlobalSubscribe}}`	`subscribed`

Example:

```
{
  "action": "setStatus",
  "destination": "{{GlobalSubscribe}}",
  "status": "subscribed"
}
```

Global unsubscribe (from all email groups)

Action	Destination	Status
setStatus	`{{GlobalSubscribe}}`	`unsubscribed`

Example:

```
{
  "action": "setStatus",
  "destination": "{{GlobalSubscribe}}",
  "status": "unsubscribed"
}
```

AppCloud decision response

Action	Destination	Status	Update All Records
setStatus	`{{DecisionInstance(<id>).Execution[<id>]}}`	`yes` / `no` / `errored` / `invalid` / `permission`	`true` / `false`

Example:

```
{
  "action": "setStatus",
  "destination": "{{DecisionInstance(da6ed61da73441dba1349582a4fa8a2c).Execution
[12345]}}",
  "status": "yes",
```

```
  "updateAll": false
}
```

AppCloud action response

Action	Destination	Status	Update All Records
setStatus	`{{ActionInstance(<id>).Execution[<id>]}}`	`active` / `complete` / `errored` / `invalid` / `permission`	`true` / `false`

Example:

```
{
  "action": "setStatus",
  "destination": "{{ActionInstance(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
  "status": "complete",
  "updateAll": true
}
```

AppCloud feeder response

Action	Destination	Status
setStatus	`{{FeederInstance[<id>]}}`	`active` / `complete` / `errored`

Example:

```
{
  "action": "setStatus",
  "destination": "{{FeederInstance(da6ed61da73441dba1349582a4fa8a2c)}}",
  "status": "complete"
}
```

AppCloud decision response (program canvas only)

Action	Destination	Status	Update All Records
setStatus	`{{DecisionInstance(<id>).Execution[<id>]}}`	`yes` / `no` / `errored` / `invalid` / `permission`	`true` / `false`

Example:

```
{
  "action": "setStatus",
  "destination": "{{DecisionInstance(da6ed61da73441dba1349582a4fa8a2c).Execution
[12345]}}",
  "status": "yes",
  "updateAll": false
}
```

AppCloud action response (program canvas only)

Action	Destination	Status	Update All Records
setStatus	`{{ActionInstance(<id>).Execution[<id>]}}`	`active` / `complete` / `errored` / `invalid` / `permission`	`true` / `false`

Example:

```
{
  "action": "setStatus",
  "destination": "{{ActionInstance(f82d50cd86a94fcab37e4ec9a98b0339).Execution
[12345]}}",
  "status": "complete",
  "updateAll": true
```

```
}
```

AppCloud Feeder Response (Program Canvas only)

Action	Destination	Status
setStatus	`{{FeederInstance[<id>]}}`	`active` / `complete` / `errored`

Example:

```
{
  "action": "setStatus",
  "destination": "{{FeederInstance(da6ed61da73441dba1349582a4fa8a2c)}}",
  "status": "complete"
}
```

Sync action reference for accounts

Add to account list

Action	Destination	Status
add	`{{AccountList[<id>]}}`	n/a

Example:

```
{
  "action": "add",
  "destination": "{{AccountList[12345]}}"
}
```

Remove from account list

Action	Destination	Status
remove	`{{AccountList[<id>]}}`	n/a

Example:

```
{
  "action": "remove",
  "destination": "{{AccountList[12345]}}"
}
```

Sync action reference for email addresses

Email addresses global unsubscribe

Action	Destination	Status
setStatus	`{{GlobalSubscribe}}`	`unsubscribed`

Example:

```
{
  "action": "setStatus",
  "destination": "{{GlobalSubscribe}}",
  "status": "unsubscribed"
}
```

Email addresses set bounceback

Action	Destination	Status
setStatus	`{{Bounceback}}`	`bounced`

Example:

```
{
  "action": "setStatus",
  "destination": "{{Bounceback}}",
  "status": "bounced"
}
```

Sync action app statuses

Here is an explanation of sync action app statuses, the status displayed, and when each status is set.

Sync action app status	Status Displayed	Set When
`active`	Being processed by app	Step records are successfully exported.
`complete`	N/A	Records have been successfully processed.
`errored`	Status 'Error' set by app	There are errors not related to configuration or Manage Data Export Permission.
`invalid`	Invalid app configuration	There is a configuration error. For example, a field with an incorrect data type has been selected to be mapped.
`permission`	Eloqua configuration error - Manage Data Export Permission	The Bulk API User does not have the "Manage Data Export" action permission. If the app encounters a 403 error when attempting to export records this would indicate the Bulk API User does not have the "Manage Data Export" action permission.

Sync status types

When you set up a sync and poll its URI,(perform a `GET` request on `https://<host>/api/bulk/2.0/syncs/<ID>`), Eloqua includes a `status` parameter in its response.

The following status values are possible:

- `pending`: The sync is queued to execute.
- `active`: The sync is currently in progress.
- `success`: The sync has completed successfully.
- `warning`: There was unexpected behavior but the sync did not fail--further details can be found by calling the `/syncs/<ID>/logs` endpoint.
- `errored`: The sync has failed. Further details can be found by calling the `/syncs/<ID>/logs` endpoint.

If you encounter an error, you may be able to gain additional information on why the sync was unsuccessful. See the Troubleshooting for more information.

Activity fields

Activity elements behave differently from other Eloqua elements. Each activity type has its own set of associated fields which are detailed below.

- EmailOpen
- EmailClickthrough
- EmailSend
- Subscribe
- Unsubscribe
- Bounceback
- WebVisit
- PageView

- FormSubmit
- ExternalActivity

For activity field type and length, see activity field details.

EmailOpen activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}",
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

EmailClickthrough activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "EmailClickedThruLink": "{{Activity.Field(EmailClickedThruLink)}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}"
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

EmailSend activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

```
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "SubjectLine": "{{Activity.Field(SubjectLine)}}",
  "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "EmailSendType": "{{Activity.Field(EmailSendType[1])}}"
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

Subscribe activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}"
}
```

[1]Possible values: Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI, SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, SecureEmailDeployment

Unsubscribe activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}"
}
```

Bounceback activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "EmailAddress": "{{Activity.Field(EmailAddress)}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
  "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
  "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
  "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
  "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
}
```

WebVisit activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
```

```
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "NumberOfPages": "{{Activity.Field(NumberOfPages)}}",
  "FirstPageViewUrl": "{{Activity.Field(FirstPageViewUrl)}}",
  "Duration": "{{Activity.Field(Duration)}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
  "WebVisitSavedId": "{{Activity.Field(WebVisitSavedId)}}"
}
```

PageView activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "WebVisitId": "{{Activity.Field(WebVisitId)}}",
  "Url": "{{Activity.Field(Url)}}",
  "ReferrerUrl": "{{Activity.Field(ReferrerUrl)}}",
  "IpAddress": "{{Activity.Field(IpAddress)}}",
  "IsWebTrackingOptedIn": "{{Activity.Field(IsWebTrackingOptedIn)}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "LinkedToContactDate": "{{Activity.Field(LinkedToContactDate)}}",
  "PageViewSavedId": "{{Activity.Field(PageViewSavedId)}}",
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

FormSubmit activity fields

```
{
  "ActivityId": "{{Activity.Id}}",
  "ActivityType": "{{Activity.Type}}",
  "ActivityDate": "{{Activity.CreatedAt}}",
  "ContactId": "{{Activity.Contact.Id}}",
  "VisitorId": "{{Activity.Visitor.Id}}",
  "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
  "AssetType": "{{Activity.Asset.Type}}",
  "AssetId": "{{Activity.Asset.Id}}",
  "AssetName": "{{Activity.Asset.Name}}",
  "RawData": "{{Activity.Field(RawData)}}",
  "FormSubmitSavedId": "{{Activity.Field(FormSubmitSavedId)}}",
  "CampaignId": "{{Activity.Campaign.Id}}",
  "ExternalId": "{{Activity.ExternalId}}",
  "CampaignResponseDate": "{{Activity.CampaignResponse.CreatedAt}}",
  "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
(MemberStatus)}}"
}
```

ExternalActivity activity fields

```
{
   "ActivityId": "{{Activity.Id}}",
   "ActivityType": "{{Activity.Type}}",
   "ActivityDate": "{{Activity.CreatedAt}}",
   "ExternalAssetName": "{{Activity.ExternalAssetName}}",
   "ExternalAssetType": "{{Activity.ExternalAssetType}}",
   "ExternalType": "{{Activity.ExternalActivityType}}",
   "CampaignId": "{{Activity.Campaign.Id}}",
   "ContactId": "{{Activity.Contact.Id}}",
   "ExternalId": "{{Activity.ExternalId}}",
   "EmailAddress": "{{Activity.Field(EmailAddress)}}",
   "ContactId": "{{Activity.Contact.Id}}",
   "CampaignId": "{{Activity.Campaign.Id}}",
   "CRMCampaignId": "{{Activity.Campaign.Field(CRMCampaignId)}}",
   "CampaignName": "{{Activity.Campaign.Field(CampaignName)}}",
   "CampaignResponseCreatedAt": "{{Activity.CampaignResponse.CreatedAt}}",
   "CampaignResponseMemberStatus": "{{Activity.CampaignResponse.Field
```

```
(MemberStatus)}}"
}
```

Activity field details

Field type	Data type	Max length	Description
`{{Activity.Id}}`	Integer (big Integer for page views)	-	The unique identifier of the activity per activity type, except for EmailSend The activity ID is not unique across activity types, or for EmailSend.
`{{Activity.Type}}`	String	100	The type of activity. The activity types are EmailOpen, EmailClickthrough, EmailSend, Subscribe, Unsubscribe, Bounceback, WebVisit, PageView, and FormSubmit.
`{{Activity.CreatedAt}}`	Date/time stamp	-	The date and time of the activity
`{{Activity.Field(EmailAddress)}}`	String	400	The email address of the contact who performed the activity
`{{Activity.Contact.Id}}`	Integer	-	The ID of the contact who performed the activity
`{{Activity.Field(EmailRecipientId)}}`	String	38	The recipient ID associated with an email activity

Field type	Data type	Max length	Description
			The recipient ID is present on every link in an email in the format of elq=*GUID*. At the time of sending the email, a GUID is generated and injected into the link. Learn more about the RecipientId.
`{{Activity.Asset.Type}}`	String	100	The type of asset associated with the activity, such as email and form
`{{Activity.Asset.Name}}`	String	100	The name of the asset associated with the activity
`{{Activity.Asset.Id}}`	Integer	-	The ID of the asset associated with the activity
`{{Activity.Field(SubjectLine)}}`	String	500	The subject line for an email activity
`{{Activity.Field(EmailWebLink)}}`	String	1000	The link for viewing the email in a web browser window for an email activity
`{{Activity.Campaign.Id}}`	Integer	-	The ID of the campaign associated with the

Field type	Data type	Max length	Description
			activity
{{Activity.Campaign.Field (CRMCampaignId)}}	String	100	The ID of the CRM campaign associated with the activity
{{Activity.Campaign.Field (CampaignName)}}	String	100	The name of the campaign associated with the activity
{{Activity.Field (EmailDeploymentId)}}	Integer	-	The ID of the email deployment for an email activity
{{Activity.Field(IpAddress)}}	String	50	The IP address of the visitor who performed the activity
{{Activity.Visitor.Id}}	Integer	-	The ID of the visitor who performed the activity
{{Activity.Visitor.ExternalId}}	String	38	The GUID of the visitor who performed the activity
{{Activity.Field (EmailClickedThruLink)}}	String	1000	The link clicked through for an EmailClickthrough activity
{{Activity.Field(RawData)}}	String	64000	The raw data submitted with a FormSubmit activity. Includes the HTML field name and the submitted values in a query string format.
{{Activity.Field(ReferrerUrl)}}	String	1000	The URL or form submission a PageView or WebVisit activity was referred from. Example form submission:

Field type	Data type	Max length	Description
			"Form Submitted: <FormName>"
{{Activity.Field(WebVisitId)}}	Integer	-	The ID of the WebVisit associated with a PageView activity
{{Activity.Field(Url)}}	String	1000	The URL viewed with a PageView activity
{{Activity.Field (IsWebTrackingOptedIn)}}	Boolean	-	Indicates if visitor who performed activity is opted-in to web tracking
{{Activity.Field(NumberOfPages)}}	Integer	-	The number of pages viewed with a WebVisit activity
{{Activity.Field(FirstPageViewUrl)}}	String	1000	The URL of the first page viewed in a WebVisit activity
{{Activity.Field(Duration)}}	String (ISO-8601 Duration)	100	The duration of a WebVisit activity returned using the ISO-8601 standard for specifying durations
{{Activity.Field(EmailSendType)}}	String	100	The type of email send for an email activity. Email send types are Batchwizard, Campaign, CDOServiceStep, DeliverabilityTest, ExternalTest, FormProcessingStep, LowVolumeAPI, ProgramBuilder, QuickSend, SOAPAPI,

Field type	Data type	Max length	Description
			SalesTools, TestCenter, Deliverability, OutlookPlugin, CampaignNotification, WelcomeEmailSender, and SecureEmailDeployment.
{{Activity.ExternalId}}	String	20	The unique identifier of the activity across activity types
{{Activity.Field(SmtpErrorCode)}}	String	9	The SMTP Status Code for the email bounceback
{{Activity.Field(SmtpStatusCode)}}	String	3	The SMTP Response Code for the email bounceback
{{Activity.Field(SmtpMessage)}}	String	510	The SMTP message for the email bounceback
{{Activity.Field (LinkedToContactDate)}}	Date and time stamp	-	The date and time the visitor was linked to the contact
{{Activity.Field(WebVisitSavedId)}}	Integer	--	The unique identifier of the web visit session.
{{Activity.Field(PageViewSavedId)}}	Integer	--	The unique identifier of the page view.
{{Activity.Field (FormSubmitSavedId)}}	Integer	--	The unique identifier of the form submission.
{{Activity.CampaignResponse.CreatedAt}}	Date/time stamp	-	The date and time the campaign response was created.
{{Activity.CampaignResponse.Field (MemberStatus)}}	String	150	The status of the campaign member.

Field type	Data type	Max length	Description
`{{Activity.ExternalAssetName}}`	String	100	The first sub-category for the asset type. For example, if it is a Tradeshow asset type, then the asset name could be the name of the Tradeshow.
`{{Activity.ExternalAssetType}}`	String	100	The top-level classification for something that your contacts are performing, for example, a Tradeshow.
`{{Activity.ExternalActivityType}}`	String	100	The specific activity that the contact or prospect performed for this specific external asset. For example, if it is a tradeshow event, then activities could include "Registered", "Canceled", "Completed", and so on.

Bulk API limits

Every Eloqua instance has the following soft limits by Bulk API sync type:

Bulk API Sync Type	Limit
Export	2000 per hour
Import	2000 per hour
Sync Action	4000 per hour

There are many variables to sync processing. We'd expect minimal impact due to volume on its own if within these limits and following best practices. The number of syncs being executed is logged and monitored to ensure the Bulk API's stability and reliability.

> **Important**: If the sync limit is exceeded, it is possible there will be slowdowns in processing imports and exports in Eloqua. In certain cases, it is possible that other clients could be affected. If there is impact on other clients, we may take action to alleviate the effect on those clients.

Import limits:

- **Definition limit**: An import definition can include a maximum of **100 fields**. The import definition will fail to create if more than 100 fields are specified.
- **Request limit**: There is a hard limit of **32 MB per request** when posting data to an import sync's staging area. Requests larger than 32 MB will fail by returning a 404 Not Found response. To import more than 32 MB of data, you can make multiple posts of 32 MB (or less) to the import definition before syncing.
- **Staging limit**: It is recommended to sync an import whenever approximately **250 MB** of data has been posted to its staging area.

Export limits

- **Definition limits**: An export definition can include a maximum of **250 fields**. The export definition will fail to create if more than 250 fields are specified.

> **Note**: When specifying fields for an app or service instance, you can specify a maximum amount of **249** fields because AppCloud developer reserves the contact ID field, which is required to update the status of records and move them to subsequent steps or pass cloud content.

- **Request limit**: There is a limit of **50,000 records** per request while retrieving data from an export. You can use the offset parameter to retrieve additional records.
- **Staging limit**: When exporting **activities** there is a limit of **5 million records** per sync.

Data limitations:

Request limit: If a POST request to a data endpoint **exceeds 32 MB**, a 404 Not Found response will be returned. Multiple post requests of 32 MB or less must be made instead. This limitation applies to following data endpoints:

- /api/bulk/2.0/accounts/syncActions/{id}/data
- /api/bulk/2.0/accounts/imports/{id}/data
- /api/bulk/2.0/activities/imports/{id}/data
- /api/bulk/2.0/contacts/syncActions/{id}/data
- /api/bulk/2.0/contacts/imports/{id}/data
- /api/bulk/2.0/customObjects/{parentId}/syncActions/{id}/data
- /api/bulk/2.0/customObjects/{parentId}/imports/{id}/data
- /api/bulk/2.0/events/{parentId}/imports/{id}/data

Field names

Every field in Eloqua has two names: the *Display Name* and the *Internal Name*. The Display Name is the name that Eloqua users see when they are editing a contact form; the internal name is a unique name for that field. For example, Contacts and accounts both have email address fields: Eloqua may use *Email Address* as the display name for both entity types, but they have distinct Internal Names.

To narrow down the list of parameters to ones that are relevant for you, use query parameters to search for the fields that you need.

For example, the following request searches for contact fields whose names begin with `Email`:

```
https://secure.p03.eloqua.com/API/Bulk/2.0/contacts/fields?q="name=Email*"
```

Important

Search terms are *case sensitive*: if you search for email, you will not find any fields, because Eloqua's field names are capitalized.

The results should resemble:

```
{
  "count": 3,
  "hasMore": false,
  "items": [
    {
```

```
      "createdAt": "1900-01-01T05:00:00.0000000Z",
      "dataType": "emailAddress",
      "hasNotNullConstraint": false,
      "hasReadOnlyConstraint": false,
      "hasUniquenessConstraint": true,
      "internalName": "C_EmailAddress",
      "name": "Email Address",
      "statement": "{{Contact.Field(C_EmailAddress)}}",
      "updatedAt": "1900-01-01T05:00:00.0000000Z",
      "uri": "/contacts/fields/100001"
    },
    {
      "createdAt": "1900-01-01T05:00:00.0000000Z",
      "dataType": "string",
      "hasNotNullConstraint": false,
      "hasReadOnlyConstraint": false,
      "hasUniquenessConstraint": false,
      "internalName": "C_EmailDisplayName",
      "name": "Email Display Name",
      "statement": "{{Contact.Field(C_EmailDisplayName)}}",
      "updatedAt": "1900-01-01T05:00:00.0000000Z",
      "uri": "/contacts/fields/100005"
    },
    {
      "createdAt": "1900-01-01T05:00:00.0000000Z",
      "dataType": "string",
      "hasNotNullConstraint": false,
      "hasReadOnlyConstraint": false,
      "hasUniquenessConstraint": false,
      "internalName": "C_SFDC_EmailOptOut1",
      "name": "SFDC Email Opt-Out",
      "statement": "{{Contact.Field(C_SFDC_EmailOptOut1)}}",
      "updatedAt": "1900-01-01T05:00:00.0000000Z",
      "uri": "/contacts/fields/100043"
    }
  ]
}
```

The `statement` field shows the Eloqua markup language representation of that field, which you can then use in an import or export definition.

System time stamps

The following system time stamps are available for contacts, accounts, custom objects / events, and activities. When setting the `areSystemTimestampsInUTC` parameter to `true` for an export definition, these are the fields that will be returned in coordinated universal time (UTC).

Contacts

System time stamp name	Internal name
Date created	`C_DateCreated`
Date modified	`C_DateModified`
Last modified by CRM system	`C_LastModifiedByExtIntegrateSystem`

Accounts

System timestamp name	Internal name
Date created	`M_DateCreated`
Date modified	`M_DateModified`
Last modified by CRM system	`M_LastModifiedByExtIntegrateSystem`

Custom objects and events

System time stamp name	Internal name
DataCard created at	`CreatedAt`
DataCard updated at	`UpdatedAt`

Activities

System timestamp name	Internal name
Activity created at	`CreatedAt`
Linked to contact date	`LinkedToContactDate`

Default display formats

For custom object fields, you can specify how date/time and numeric fields are formatted. These display formats can be managed as needed in Eloqua if you need to alter or create new formats. The following table outlines the default display formats and their associated ids. See an example where FieldOutputFormat is set.

Date Formats

Format	Example	id
d/M/yyyy	15/6/2009	17
dd/MM/yy	24/06/09	7
dd/MM/yyyy	24/06/2009	6
dddd	Wednesday	11
dddd MMMM d, yyyy	Wednesday, September 9, 2009	19
h.mm.ss	2.30.04	3
h:mm:ss tt	2:28:46 AM	15
HH tt	14 PM	13
HH.mm.ss	14.28.46	2
HH:mm	14:28	12
HH:mm:ss	14:28:46	14
M.d.yyyy	6.15.2009	1
M/d/yyyy	6/15/2009	18
MM/dd/yy	06/24/09	5
MM/dd/yyyy	06/24/2009	4
MMM d, yyyy	Aug 20, 2009	16
MMMM d, yyyy	August 20, 2009	8

Format	Example	id
MMMM, d	August, 20	9
MMMM, yyyy	August, 2009	10
"yyyy-MM-dd HH:MM:ss.fff"	2004-03-27 14:03:04.000	26

Numeric Formats

Format	Example	id
#	12345	22
#,#.##	12,345.68	23
#,0	12,346	27
#.#	12345.7	21
#.##	12345.68	20
#.####	12345.6789	25
#.0	12345.0	28
0.0	12345.0	29

Bulk API data types

When using the bulk API to retrieve contact and custom object fields, the bulk API returns the following data types:

- `emailAddress`
- `string`
- `date`
- `number`

However the bulk API does not provide insight into the granularity of the data type. For example, the `string` and `number` data types could refer to more granular specifications that can be useful for building integrations

Example: A `string` data type could be a small text or large text field. The difference between these two data types is significant when it comes to character limits, e.g. small text is 100 for contact fields and 250 for custom object fields; whereas, large text is 32,000 for both contact and custom object fields.

For custom objects, a `number` data type could be a numeric or number field. The difference between these two data types is significant as it is the difference of an integer versus a float.

Determining data type granularity

Using the application API endpoints, developers can retrieve contact and custom object fields to see more details about the data type. You can use the following endpoints to retrieve fields using the application API.

Application API fields endpoints

Eloqua entity	Endpoint
Contacts	Retrieve a contact field
Contacts	Retrieve a list of contact fields (depth set to complete)
Custom objects	Retrieve a custom object
Custom objects	Retrieve a list of custom objects (depth set to complete)

Example

Retrieve contact fields using the bulk API.

Request

```
GET api/bulk/2.0/contacts/fields?limit=2
```

Response


```
{
  "items": [
    {
      "name": "Department",
      "internalName": "C_Department1",
      "dataType": "string",
      "defaultValue": "Hardware",
      "hasReadOnlyConstraint": false,
      "hasNotNullConstraint": false,
      "hasUniquenessConstraint": false,
      "statement": "{{Contact.Field(C_Department1)}}",
      "uri": "/contacts/fields/100211",
      "createdBy": "API.User",
      "createdAt": "2018-08-17T12:09:00.0000000Z",
      "updatedBy": "API.User",
      "updatedAt": "2018-08-17T12:33:20.0270000Z"
    },
    {
      "name": "Large Text Field",
      "internalName": "C_Large_Text_Field1",
      "dataType": "string",
      "hasReadOnlyConstraint": false,
      "hasNotNullConstraint": false,
      "hasUniquenessConstraint": false,
      "statement": "{{Contact.Field(C_Large_Text_Field1)}}",
      "uri": "/contacts/fields/100212",
      "createdBy": "API.User",
      "createdAt": "2018-09-26T19:16:00.0000000Z",
      "updatedBy": "API.User",
      "updatedAt": "2018-09-26T19:16:00.0000000Z"
    }
  ],
  "totalResults": 82,
  "limit": 2,
```

```
  "offset": 0,
  "count": 2,
  "hasMore": true
}
```

Notice the `dataType` for both of these contact fields are `string`.

Using the application API to retrieve these contact fields, we can get more information about the data types.

Now let's retrieve contact fields using the application API with `depth` set to complete.

Request

```
GET api/rest/1.0/assets/contact/fields?depth=complete&count=2
```

Response

```
{
   "elements": [
      {
         "type": "ContactField",
         "id": "100211",
         "createdAt": "1534507740",
         "createdBy": "35",
         "depth": "complete",
         "name": "Department",
         "updatedAt": "1534509200",
         "updatedBy": "35",
         "dataType": "text",
         "defaultValue": "Hardware",
         "displayType": "singleSelect",
         "internalName": "C_Department1",
         "isReadOnly": "false",
         "isRequired": "false",
         "isStandard": "false",
         "optionListId": "66",
```

```
        "isPopulatedInOutlookPlugin": "false",
        "updateType": "always"
      },
      {
        "type": "ContactField",
        "id": "100212",
        "createdAt": "1537989360",
        "createdBy": "9",
        "depth": "complete",
        "name": "Large Text Field",
        "updatedAt": "1537989360",
        "updatedBy": "9",
        "dataType": "largeText",
        "displayType": "text",
        "internalName": "C_Large_Text_Field1",
        "isReadOnly": "false",
        "isRequired": "false",
        "isStandard": "false",
        "isPopulatedInOutlookPlugin": "true",
        "updateType": "always"
      }
    ],
    "page": 1,
    "pageSize": 2,
    "total": 82
}
```


Notice the `dataType` for the first contact field is `text`, but the second is `largeText`. The data type is important when considering character limits. See the Eloqua help center for information about data types and custom object field types.

Troubleshooting

Eloqua's bulk API includes endpoints to help you troubleshoot your imports and exports. Specifically, these endpoints provide insight into what happened during the sync phase of an import or export.

To control for different data access permissions, Eloqua associates imports and exports with the user who creates them. Only the user who synchronizes an export is able to retrieve the exported data: any other user who attempts to retrieve it will receive an error.

To retrieve data for a given export definition, resync that data yourself and then retrieve it.

View sync logs

Sync logs provide aggregate data detailing what happened during a specific sync. Unlike the sync's `status` field, which simply indicates success, failure, and errors, the `/syncs/{id}/logs` endpoint response includes the Eloqua status codes and a human-readable message.

For example, calling `https://<host>.eloqua.com/API/Bulk/2.0/syncs/1196/logs` yields:

```
{
  "count": 3,
  "hasMore": false,
  "items": [
    {
      "count": 4149,
      "createdAt": "2014-05-12T14:23:05.9100000Z",
      "message": "Successfully exported members to csv file.",
      "severity": "information",
      "statusCode": "ELQ-00102",
      "syncUri": "/syncs/1196"
    },
```

```
      {
         "count": 0,
         "createdAt": "2014-05-12T14:23:07.2670000Z",
         "message": "Deleted internal staging for data transfer.",
         "severity": "information",
         "statusCode": "ELQ-00131",
         "syncUri": "/syncs/1196"
      },
      {
         "count": 0,
         "createdAt": "2014-05-12T14:23:07.7630000Z",
         "message": "Successfully converted csv file to sqlite, taking 264 kb.",
         "severity": "information",
         "statusCode": "ELQ-00106",
         "syncUri": "/syncs/1196"
      }
   ],
   "limit": 1000,
   "offset": 0,
   "totalResults": 3
}
```


This provides more granular information and can help you debug where an unsuccessful sync went wrong.

Check for import errors

The `/syncs/{id}/rejects` endpoint gives raw data about validation failures during **imports**.

Eloqua performs validation on some data. If you attempt to import a non-valid email address using the bulk API, that record will be rejected. The `/syncs/{id}/rejects` endpoint allows you to see what records were not imported, so that you can correct any issues and try again.

In some cases, you may choose to reuse an export definition. For example, if you are performing routine exports, reusing an existing definition can save time and effort. However, when you resync the export, the exported data may have changed. Last week's output from `/contacts/exports/123/data` may be different from today's.

The `/syncs/{id}/data` endpoint enables you to retrieve the data associated with a *specific sync*. This can be useful to retrieve past data or debug issues retroactively.

Bulk API frequently asked questions

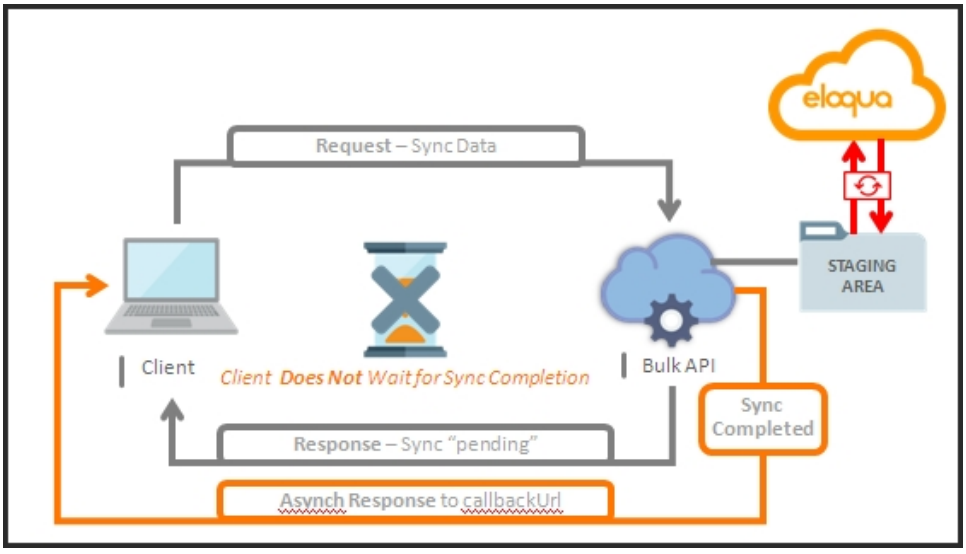
General

What trims/versions of Eloqua is the bulk API available?

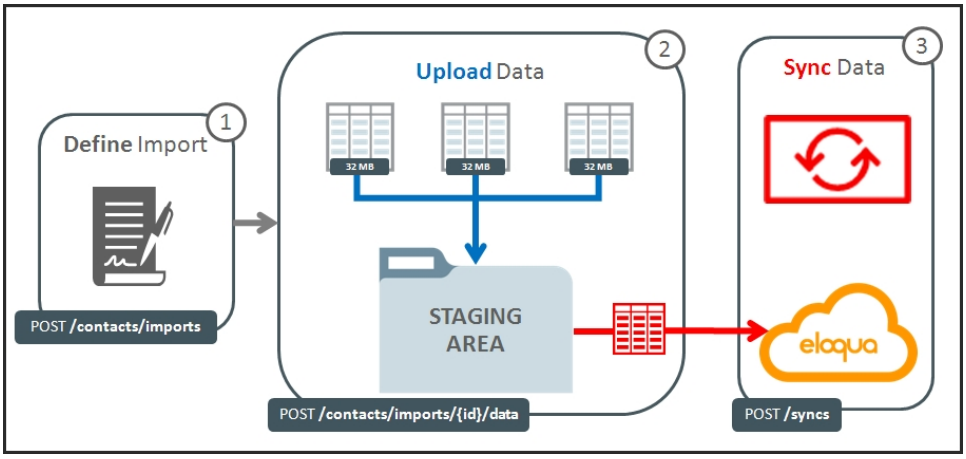
The bulk API is available on ALL trims and ALL versions of the API (E9 and E10, and Express, Team and Enterprise)

I am having trouble visualizing the Bulk API pattern? Are there any visual diagrams?

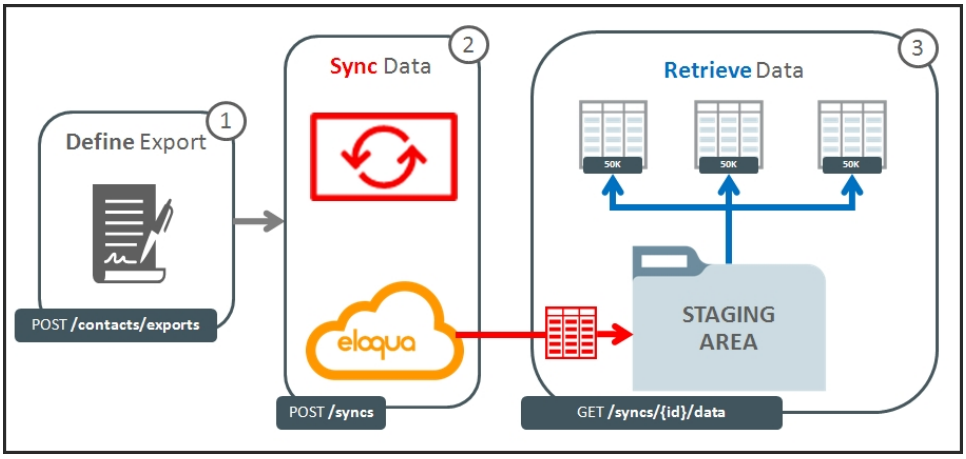
Asynchronous Flow



Importing Data



Exporting Data



Is the API rate limited?

Every Eloqua instance has the following soft limits by Bulk API sync type:

Bulk API Sync Type	Limit
Export	2000 per hour
Import	2000 per hour
Sync Action	4000 per hour

Here are the Bulk API limits:

- The amount of data allowed on our servers in the staging area
    - The amount is based on the # of contacts allowed for the client x 50 in KB; So, for a client with 10 M contacts, this would result in 500 GB storage area for that client
- Import Limits
    - Definition limit: An import definition can include a maximum of 100 fields
    - Request limit: There is a hard limit of 32 MB per request when posting data to an import sync's staging area

        - Staging limit: It is recommended to sync an import whenever approximately 250 MB of data has been posted to its staging area

- Export Limits
    - Definition limits: An export definition can include a maximum of 250 fields
    - Request limit: There is a limit of 50,000 records per request while retrieving data from an export
    - Staging limit: When exporting activities there is a limit of 5 million records per sync.

Read more about limits.

Does the bulk API provide the ability to export contact data of any given set of fields based on some filter or other requirement?

Yes; an export is limited to 250 fields at once, but any of the contact fields are available at export as are the linked account fields. In addition, Contact attributes (subscription and bounceback status) are available.

For filtering, data can be exported from a contact list, filter or segment, or based on un/subscription to an email group, a cloud connector status or comparison of a single field on the contact. See Filtering for more information.

How do you retrieve all contacts that are in a contact list?

Export contacts based on their presence in the contact List with ID 123:

```
{
  "name": "List 123 Contacts Export",
  "fields": {
    "ID": "Contact.Id",
```

```
    "FirstName": "Contact.Field(C_FirstName)",
    "LastName": "Contact.Field(C_LastName)",
    "EmailAddress": "Contact.Field(C_EmailAddress)"
  },
  "filter": "EXISTS('ContactList[123]')"
}
```

See Filtering for more information on filtering.

Can you import a bunch of contacts with an arbitrary set of fields?

Yes, imports are also defined with any set of contact fields / attributes as are exports.

Can you import into a (contact) group/list?

Yes; similar to export filtering, except that you specify additions to a contact list by using a Sync Action, with the action = "add" and the `destinationUri` as the contact list's uri (ie "/contact/list/123"). See for Sync actions more information.

Can you import into a canvas?

You can create a Feeder Service to enable importing directly into a canvas.

Another option for importing contacts into a canvas is using a Bulk API sync action to add to list. The contact list could be used in a segment that gets re-evaluated which then would add them to a canvas.

Can you unsubscribe contacts from all future mailings (when importing)?

There are two options here:

Set the contact's global subscription status to unsubscribed. You interact with this almost like any other field (ie as a column in the import).

On import, use a sync action to unsubscribe from a particular email group.

What is the maximum number of fields that can be included in an export?

250

> **Note**: When specifying fields for an app or service instance, you can specify a maximum amount of **249** fields because AppCloud developer reserves the contact ID field, which is required to update the status of records and move them to subsequent steps or pass cloud content.

I am coming close to the Bulk API syncs soft limit, or am already over this limit for one or more sync types. Am I going to get shut down? What's the impact of going over?

We want to encourage app/Bulk API usage, as long as app/Bulk API usage is not generating abusive API behavior. We advise not getting too caught up with the soft limit for Bulk API syncs, unless going over significantly, and/or observing performance issues related to the Bulk API load.

If you are going over this limit for one or more sync types, the chances of performance issues will increase with volume (meaning the higher you exceed the limit, the higher probability of experiencing performance issues). It's strongly recommended to stay as

close to the soft limits as possible, to reduce the chance of performance issues. The main impact you'll see as the sync volume increases - is increased sync processing time with increased probability of performance issues. Performance issues are more likely to occur when there is a large Bulk API sync volume with other Eloqua activities (such as Lead Scoring, Form Processing, Segment Execution, and so on) occurring concurrently and/or also in large volumes. The lower the Bulk API sync volume, the smaller probability of performance issues when these other activities are occurring.

I am exporting all of my activities using the Bulk API. When I accumulate activities exported via the Bulk API, metrics are not matching what I see an Insight. Why wouldn't they match?

The are two primary reasons they don't match:

1. Any data pulled from APIs does not include activity from deleted Contacts. Insight includes activity from deleted Contacts.

2. Label-based Access Control (LBAC), if it's being used. The APIs respect LBAC, so sometimes the user placing the API request doesn't have access to view Contacts, or the User doesn't have access to view Contacts in Insight. Additionally, non contact related reports in Insight are not limited to the contacts that a user can access. For example, a campaign report will show data related to all campaign members.

Are there any best practices or resources for exporting all activities via the Bulk API?

Exporting All Eloqua Activities Using the Bulk API: Flowchart, Best Practices, and Resources

Is it possible the Bulk API would export duplicate records?

Yes, if you use `UpdatedAt` in the definition's filter the same record may be returned more than once if a record is updated during the export. This applies to all Bulk API exports that allow `UpdatedAt` in the filter.

Reporting API

Oracle Eloqua's reporting API is a RESTful API that provides access to data from the Insight data warehouse.

Getting Started: Learn the key concepts for getting started with the reporting API; the API call format, query options, and more.

- Getting started with the reporting API
- Query options overview
- Pagination

Best Practices: Learn best practices for optimizing your use of the Reporting API.

- Reporting API best practices
- Reporting API FAQ

Endpoints: Detailed information on endpoint definitions and their parameters.

- API reference

Getting started with the reporting API

Oracle Eloqua's reporting API allows you to access data directly from the Insight data warehouse. The reporting API is built using Open Data Protocol (OData), a REST-based open web protocol for sharing data in a standardized form for easy consumption. OData is different from other REST-based web services in that it provides a uniform way to describe both the data and the data model. Eloqua's reporting API is

recommended for use when you want to retrieve reporting data directly from the Insight data warehouse.

Considerations

To use the reporting API, you will need to ensure that you have an Eloqua instance and that your user has an account on this instance. Additionally, the user will need to be assigned the analyzer license. This ensures that customers have precise control over access permissions to the reporting API.

Some Eloqua instances enable contact-level security, which restricts access to data based on different user roles. For example, users might only have access to contacts located in their geographical region. Because these permissions affect what data the user can access, it is important that the Eloqua user accessing the API has the appropriate permissions. Learn more.

API Call Format

The reporting API call consists of an initial authentication and the following API call format. The reporting API only supports GET commands.

```
GET [baseURL]/API/odata/[subjectArea]/[APIVersionIdentifier]/[endpoint]
```

Limits

The reporting API has a maximum limit of returning 10,000 records per page. Utilize the `$top` parameter to specify the number of records returned and implement pagination for larger datasets. By default, the reporting API will return a maximum of 25 months of historical data, unless otherwise specified.

Request soft limit: 1,000 per hour.

Processing limits are influenced by various factors. If operations remain within these specified limits and adhere to best practices, the impact of volume alone is expected to be minimal. We continuously log and monitor the number of API requests executed to uphold stability and reliability.

Query options overview

The Oracle Eloqua reporting API offers a comprehensive query language that can be appended to GET requests, enabling the retrieval of specific data elements. These query options are flexible and can be combined as needed. Below are some of the commonly used OData query options that are supported by the reporting API, unless otherwise noted.

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

$filter

Filters a query to only include items that match the specified expressions. For example, the following campaign query filters the retrieval campaigns with a Completed status.

```
GET /api/odata/campaignAnalysis/1.0/campaign?$filter=campaignStatus eq 'Completed'
```

$select

Specifies a limited subset of properties to include in the results. Multiple properties may be specified as a comma-separated list. For example, the following query returns only the total sends, delivered, opens and clickthroughs email activities.

```
GET
/api/odata/campaignAnalysis/1.0/emailActivities?$select=eloquaCampaignId,totalSends,totalDelivered,totalOpens,totalClickthroughs
```

$count

Returns the number of records in a collection or if the collection has a filter, the number of records matching the filter. For example, the following query returns email activities associated to campaigns and returns a count of the records returned. Since email activities are aggregated by several columns, the query counts by those aggregating columns.

```
GET /api/odata/campaignAnalysis/1.0/emailActivities?$filter=eloquaCampaignId gt 0&$count=true
```

$orderBy

Sorts the results according to the specified property and in the specified direction. For example, the following query returns form submission activities sorted by the submission date and hour, starting with the most recent date (descending).

```
GET /api/odata/campaignAnalysis/1.0/formActivities?$filter=dateHour gt 2022-01-01&$orderby=dateHour desc
```

Pagination

Pagination is an effective technique for efficiently managing and retrieving large datasets in the reporting API.

Pagination in OData

The reporting API is built on the OData protocol, which provides standardized mechanisms for implementing pagination. This allows developers to fetch data in manageable chunks. The key components of pagination in OData are the `$skip` and `$top` query options. These options enable developers to specify the number of items to skip and the maximum number of items to return per page, respectively.

Using pagination

To navigate through multiple pages of data, we utilize the `$skip` and `$top` query options. For example:

```
GET /api/odata/campaignAnalysis/1.0/emailActivities?$skip=100&$top=100
```

`$skip=100`: This parameter instructs the server to skip the first 100 records in the dataset, facilitating navigation to a specific page of results.

`$top=100`: This parameter specifies that the server should return a maximum of 100 records in the response, defining the size of each page of data returned by the API.

Handling pagination response

Upon receiving the request, the server responds with a paginated collection of email activities based on the specified pagination parameters. Additionally, the response may include metadata indicating the total count of records and links to navigate to the next and previous pages. By implementing pagination effectively, developers can

optimize data retrieval processes, enhance performance, and ensure a seamless user experience when interacting with the reporting API.

Reporting API best practices

Query modified records based on last execution

Improve data retrieval efficiency by querying only modified records since your last execution. This focused approach — filtering queries to include only modified records — reduces data transfer and processing overhead, resulting in faster operations.

Use pagination to manage large volumes

Pagination facilitates the retrieval of data in manageable portions. In OData, the essential elements for pagination are the `$skip` and `$top` query options. These parameters empower developers to dictate both the quantity of records to bypass and the maximum number of records to present per page. Pagination is especially advisable when dealing with high data volumes.

Utilize `$select` to retrieve only necessary properties

By default, queries return all available properties of an entity. Adding the `$select` query allows you to specify only the properties you need and improves performance.

In the following example, using the `$select` query, we will only return the campaign ID, total sends, total delivered, total opens, and total clickthrough properties:

```
GET/api/odata/campaignAnalysis/1.0/emailActivities?$select=eloquaCampaignId,totalSends,totalDelivered,totalOpens,totalClickthroughs
```

Derive calendar dimensions with the `dateHour` key

The reporting API includes a calendar dimension as part of each endpoint subject area. The primary key is `dateHour`, which can be used to join with activity endpoints to derive additional calendar dimensions.

The calendar table is useful for reporting because it can help determine business days, fiscal quarters, and so on. For example, you can use this endpoint to map `dateHour` activity details to the calendar day, month, quarter, year, and so on.

Understand the relationship between the reporting API and Insight subject areas

The reporting API endpoints are closely aligned with the subject area structure found in Insight. A subject area implies the data level to which metrics and attributes relate. It is highly recommended to familiarize yourself with Insight reporting and the data sources available through subject areas.

Note that not all subject areas will be launched and supported by the reporting API. Given the extensive range of metrics and attributes within Insight's subject areas, achieving complete parity with the reporting API will take time and require ongoing releases.

Reporting API FAQ

Does the reporting API respect Label-Based Access Control (LBAC)?

Yes, LBAC is applied to the reporting API; ensuring that the data returned is based on the user's authentication and query privileges.

Is there reporting API available on all trims?

Yes, the reporting API is available on all trims.

How do I get access to the reporting API?

To access the reporting API, you'll need a user account with an assigned Analyzer license.

What time zone is used in the reporting API?

Unless specified otherwise, the timezone offset of '-4:00' denotes Eastern Daylight Time (EDT) for this timestamp.

What is the difference between EloquaLinkedAccountID and Account ID?

You can link contacts to accounts by a specific contact field, this relationship is configured using Account Linkage. The `eloquaLinkedAccountID` property reflects the account ID associated with the contact, as determined by rule set up in the account linkage. The `accountId` uses the contacts company field to simulate an account to contact linkage to support Fusion Marketing integrations and account level reporting.

Why does my campaign startdate show as 1900-01- 01T00:00:00- 05:00

A start date of 1900-01- 01T00:00:00- 05:00 indicates that there is no start date.

(Mac user) Why don’t I see the ODATA connection available in Excel?

Mac users need to ensure they are on version Microsoft Excel version 16.9+.

Does the reporting API provide endpoints for all subject areas?

Not all subject areas will be launched and supported by the reporting API. Given the extensive range of metrics and attributes within Insight's subject areas, achieving complete parity with the reporting API will take time. We anticipate it will require a couple of releases to fully develop the API.

Can you create import definitions?

No, at this time only GET commands are supported.

When I accumulate activities exported via the Bulk API and compare with results pulled from the reporting API they do not match. Why wouldn't they match?

It is likely you will have a difference in the Eloqua reporting API and Bulk API; the reporting API will contain metrics and activities on deleted contacts.

Changelog

View changes for Eloqua's APIs including:

- New features
- Significant recent changes
- Platform notices

Included in these changelogs are the Application API, Bulk API, and AppCloud Developer Framework.

To learn more about recent and upcoming Oracle Eloqua release rollouts, including rollout dates, user-facing features, and product notices, visit the Oracle Eloqua Release Center.

Release 24B

Bulk API

- When attempting to delete data using the data endpoints when there is an associated sync in progress, we will now return a “409 There was a conflict.” validation response. For more information, see the product notice.
- Resolved an issue where if both *IsBounced* and *IsSubscribed* are set to 1 on Bulk API contact import only *IsSubscribed* would successfully be set to 1.

Application API

- *CampaignId* added to *fieldValues* for SMS activities for the Application API 2.0 Retrieve a list of activities endpoint. For more information, see the product notice.

Reporting API

- The reporting API is now Generally Available. Learn more.

Platform notices

- With the arrival of Eloqua release 24C (August 2024), there will be a limit of 500 files per Bulk API import / sync action sync. For more information, see the product notice.

Documentation enhancements of note

- New best practice for exporting contacts in a Segment / Shared Filter added to Bulk API Best Practices.
- Details on the limit for searching custom object data added to the Using the search URL parameter tutorial.
- Details on the authorization flow limit added to Authenticate using OAuth 2.0.
- Examples using *IsBounced* and *IsSubscribed* in a Bulk API export definition filter added to Filtering.

Release 24A

Bulk API

- The Bulk API default `dataRetentionDuration` for exports will now be 6 hours. For more information, see the product notice.
- New bounceback sync action available with Bulk API email address import definition which enables changing the bounceback status of an email address without having to create a contact. An email address can be set to unsubscribed or bounced but not both. For more

information, see Sync actions.

- All CSV exports, including the Bulk API, will escape formulas by prepending each cell field with a formula with a single quote. For more information, see the product notice.

Application API

- GET requests using the Application API will be controlled by the new form action permissions for the following endpoints:
    - 2.0 Endpoints
        - Retrieve processed form data for a single form: GET /API/REST/2.0/data/form/{id}
        - Retrieve all spam form submit jobs for a single form: GET /API/REST/2.0/data/form/{formid}/formSpamData
        - Retrieve all form submit jobs for a single form: GET /API/REST/2.0/data/form/{formid}/formData
        - Retrieve a single form submit job: GET /API/REST/2.0/data/form/{formid}/formData/{formSubmitJobId}
        - Retrieve batch form submission details by batchId: GET /API/REST/2.0/data/form/{formid}/batch/formData/{batchid}
        - Retrieve batch form submission details by batchCorrelationId: GET /API/REST/2.0/data/form/{formid}/batchCorrelation/formData/{batchcorrelationid}
    - 1.0 Endpoints
        - Retrieve a single form submit job: GET /API/REST/1.0/data/form/{formId}/formData/{formSubmitJobId}
        - Retrieve all form submit jobs for a single form: GET /API/REST/1.0/data/form/{formId}/formData

- Retrieve processed form data for a single form: GET /API/REST/1.0/data/form/{id}

Release 23D

Application API

- Application API Campaign endpoints will now include `campaignClassification` at minimal depth
- Application API canvas create and update endpoints will now return a more detailed validation error when `evaluateNoAfter` is not valid
- Application API Retrieve an email endpoint will now return the destination links for Email Components within the `htmlContent` properties `html` and `htmlBody` when the `noMergeContent` query parameter is set to false
- Application API Form endpoints to include new properties when form spam protection is enabled and `isFormSpamProtection` is set to "`true`":
  - `isAutoClickProtectionEnabled`
  - `isCustomHoneypotHtmlNameEnabled`
  - `formSpamTimeInterval`
- More information is available within our Product Notices in the Oracle Marketing Developer Tools area within the Cloud Customer Connect community.
- New/Updated Application API endpoints for SMS are now available:
  - SMS Asset Management
    - `POST /API/REST/2.0/assets/sms`
    - `PUT /API/REST/2.0/assets/sms/{id}`

- `GET /API/REST/2.0/assets/sms/{id}`
- `GET /API/REST/2.0/assets/sms`
- `DELETE /API/REST/2.0/assets/sms/{id}`

- SMS Folder Management
  - `POST /API/REST/2.0/assets/sms/folder`
  - `GET /API/REST/2.0/assets/sms/folder/{id}`
  - `PUT /API/REST/2.0/assets/sms/folder/{id}`
  - `DELETE /API/REST/2.0/assets/sms/folder/{id}`
  - `PATCH /API/REST/2.0/assets/sms/folder/{id}/contents`
  - `GET /API/REST/2.0/assets/sms/folder/root`
  - `GET /API/REST/2.0/assets/sms/folder/{id}/contents`
  - `GET /API/REST/2.0/assets/sms/folder/root/contents`
  - `GET /API/REST/2.0/assets/sms/folder/root/folders`
  - `GET /API/REST/2.0/assets/sms/folder/{id}/folders`
  - `GET /API/REST/2.0/assets/sms/folders`

- SMS Setup
  - View Codes
    - `GET /API/REST/2.0/assets/sms/senderCodes`
  - Manage Keywords
    - `POST /API/REST/2.0/assets/sms/keyword`
    - `GET /API/REST/2.0/assets/sms/keywords`
    - `GET /API/REST/2.0/assets/sms/keyword/{id}`
    - `DELETE /API/REST/2.0/assets/sms/keyword/{keywordId}`
  - Manage Invalid Keyword Messages
    - `POST /API/REST/2.0/assets/sms/invalidResponseMessage`
    - `GET /API/REST/2.0/assets/sms/invalidResponseMessages`
    - `GET /API/REST/2.0/assets/sms/invalidResponseMessage/{id}`

        - `PUT /API/REST/2.0/assets/sms/invalidResponseMessage/{id}`
        - `DELETE /API/REST/2.0/assets/sms/invalidResponseMessage/{id}`

    - A new parameter, `isSmsSubscribed`, is added to the existing Contact API retrieve contact information 2.0 endpoint. This gives a contact's global subscription status for SMS.

    - New Activities API 2.0 endpoints

        - `GET /api/rest/2.0/data/activities/contact/{id}/{type}`, retrieve a list of activities of a contact including SMS activities. Learn more.

    - New Phone Number consent Management API 2.0 endpoints

        - `POST /API/REST/2.0/data/sms/subscription (X-HTTP-Method-Override=SEARCH)`, a maximum of 10 phone numbers can be passed in a single request
        - `PUT /API/REST/2.0/data/sms/subscription`

    - Campaign 2.0 endpoints now support SMS steps

    - Contact segment 2.0 endpoints now support SMS based filters

    - Security group 2.0 endpoints now support SMS related settings

Bulk API

New/Updated Bulk API endpoints for SMS are now available:

- We've introduced new phone Number consent import endpoints. You can import phone numbers & it's consent (Opt-In/Opt-Out) using it, the fields are: `{{PhoneNumber.Field(PhoneNumber)}}`, `{{PhoneNumber.Field(Preference)}}`. `{{PhoneNumber.Field(Preference)}}` must be set to `Opt-Out` or `Opt-In`, if there are duplicates, e.g. the Phone Number is repeated, the duplicates will be rejected. Phone numbers must be valid, otherwise they will be rejected.

Below are the new endpoints:

- Create a phone number import definition (`POST /api/bulk/2.0/phoneNumbers/imports`)
- Update existing phone number import definition (`PUT /api/bulk/2.0/phoneNumbers/imports/{id}`)
- Retrieve a list of phone number import definitions (`GET /api/bulk/2.0/phoneNumbers/imports`)
- Retrieve a single phone number import definition (`GET /api/bulk/2.0/phoneNumbers/imports/{id}`)
- Delete phone number import definition (`DELETE /api/bulk/2.0/phoneNumbers/imports/{id}`)
- Upload data to phone number definition (`POST /api/bulk/2.0/phoneNumbers/imports/{id}/data`)
- Delete data from phone number definition (`DELTE /api/bulk/2.0/phoneNumbers/imports/{id}/data`)

- Phone number consent & SMS Global Subscription fields are now included on Contact export endpoints.
  - `{{Contact.SMS.IsOptedIn}}`
  - `{{Contact.SMS.MobilePhone.IsOptedIn}}`
  - `{{Contact.SMS.BusPhone.IsOptedIn}}`

  The possible field values are true or false. You can also use the fields for filtering.

Release 23C

Bulk API

- External Activities are available to export via Bulk API - one of our top Dream It idea requests!

- New Activity Type: ExternalActivity
- ExternalActivity Fields:
    - Activity.Id
    - Activity.Type
    - Activity.CreatedAt
    - Activity.ExternalAssetType
    - Activity.ExternalAssetName
    - Activity.ExternalActivityType
    - Activity.Campaign.Id
    - Activity.Contact.Id
    - Activity.ExternalId
- Use the following request to retrieve all available fields for External Activities:
    - `GET /api/Bulk/2.0/activities/fields?activityType=ExternalActivity`

  Learn more about the export definition.

- Bulk API definition deletion when App uninstalled. For more information, see the product notice.

Application API

- New Lookup Table Application API endpoints:
    - Retrieve
        - `GET /api/rest/2.0/assets/lookupTables`
        - `GET /api/rest/2.0/assets/lookupTable/{id}`
        - `GET /api/rest/2.0/data/lookupTable/{parentId}/lookupTableEntries`
    - Update
        - `PUT /api/rest/2.0/assets/lookupTable/{parentId}/LookupTableEntry/{key}`

    - Delete
        - `DELETE /api/rest/2.0/assets/lookupTable/{parentId}/LookupTableEntry/{key}`

- A new property, `isExitHistoryDisabled`, will be added to the Application API Campaign and Program endpoints. For more information, see the product notice.

Release 23B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 23B.

Application API

API endpoints now available to manage asset folders:

- Campaign folders
    - POST /API/REST/2.0/assets/campaign/folder
    - GET /API/REST/2.0/assets/campaign/folders
    - PUT /API/REST/2.0/assets/campaign/folder/{id}
    - GET /API/REST/2.0/assets/campaign/folder/{id}
    - DELETE /API/REST/2.0/assets/campaign/folder/{id}
    - PATCH /API/REST/2.0/assets/campaign/folder/{id}/contents
    - GET /API/REST/2.0/assets/campaign/folder/{id}/contents
    - GET /API/REST/2.0/assets/campaign/folder/{id}/folders
    - GET /API/REST/2.0/assets/campaign/folder/root/folders
    - GET /API/REST/2.0/assets/campaign/folder/root/contents
    - GET /API/REST/2.0/assets/campaign/folder/root

- Email folders
    - POST /API/REST/2.0/assets/email/folder
    - GET /API/REST/2.0/assets/email/folders
    - PUT /API/REST/2.0/assets/email/folder/{id}
    - GET /API/REST/2.0/assets/email/folder/{id}
    - DELETE /API/REST/2.0/assets/email/folder/{id}
    - PATCH /API/REST/2.0/assets/email/folder/{id}/contents
    - GET /API/REST/2.0/assets/email/folder/{id}/contents
    - GET /API/REST/2.0/assets/email/folder/{id}/folders
    - GET /API/REST/2.0/assets/email/folder/root/folders
    - GET /API/REST/2.0/assets/email/folder/root/contents
    - GET /API/REST/2.0/assets/email/folder/root
- Form folders
    - POST /API/REST/2.0/assets/form/folder
    - GET /API/REST/2.0/assets/form/folders
    - PUT /API/REST/2.0/assets/form/folder/{id}
    - GET /API/REST/2.0/assets/form/folder/{id}
    - DELETE /API/REST/2.0/assets/form/folder/{id}
    - PATCH /API/REST/2.0/assets/form/folder/{id}/contents
    - GET /API/REST/2.0/assets/form/folder/{id}/contents
    - GET /API/REST/2.0/assets/form/folder/{id}/folders
    - GET /API/REST/2.0/assets/form/folder/root/folders
    - GET /API/REST/2.0/assets/form/folder/root/contents
    - GET /API/REST/2.0/assets/form/folder/root

- Hyperlink folders
    - POST /API/REST/2.0/assets/hyperlink/folder
    - GET /API/REST/2.0/assets/hyperlink/folders
    - PUT /API/REST/2.0/assets/hyperlink/folder/{id}
    - GET /API/REST/2.0/assets/hyperlink/folder/{id}
    - DELETE /API/REST/2.0/assets/hyperlink/folder/{id}
    - PATCH /API/REST/2.0/assets/hyperlink/folder/{id}/contents
    - GET /API/REST/2.0/assets/hyperlink/folder/{id}/contents
    - GET /API/REST/2.0/assets/hyperlink/folder/{id}/folders
    - GET /API/REST/2.0/assets/hyperlink/folder/root/folders
    - GET /API/REST/2.0/assets/hyperlink/folder/root/contents
    - GET /API/REST/2.0/assets/hyperlink/folder/root
- Image folders
    - POST /API/REST/2.0/assets/image/folder
    - GET /API/REST/2.0/assets/image/folders
    - PUT /API/REST/2.0/assets/image/folder/{id}
    - GET /API/REST/2.0/assets/image/folder/{id}
    - DELETE /API/REST/2.0/assets/image/folder/{id}
    - PATCH /API/REST/2.0/assets/image/folder/{id}/contents
    - GET /API/REST/2.0/assets/image/folder/{id}/contents
    - GET /API/REST/2.0/assets/image/folder/{id}/folders
    - GET /API/REST/2.0/assets/image/folder/root/folders
    - GET /API/REST/2.0/assets/image/folder/root/contents
    - GET /API/REST/2.0/assets/image/folder/root

Release 23A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 23A.

Application API

- API endpoints now available to manage User Security Groups:
    - GET /system/security/group/{id}/users
    - PATCH /system/security/group/{id}/users
    - DELETE /system/security/group/{id}
    - PUT /system/security/group/{id}
    - GET /system/security/group/{id}
    - POST /system/security/group
    - GET /system/security/groups
- Retrieve visitor data change: When retrieving visitor data using the Retrieve visitor data Application API 2.0 endpoint `visitorId` will now be a BigInt. Previously, `visitorId` was an Integer.
- Eloqua Form Application API changes: When a form is created, updated, or retrieved using the Application API 2.0 endpoints, the response body will return all mapped and unmapped fields present on the form for the processing steps to update Contacts, Accounts, and Custom Objects with form data. Previously, only fields included within the processing steps when the form was created or updated were returned by the response body. For more information, see the product notice.

  The following Application API 2.0 processing step endpoints are impacted:

- Create a form `POST /api/rest/2.0/assets/form`
- Retrieve a form `GET /api/rest/2.0/assets/form/{id}`
- Update a form `PUT /api/rest/2.0/assets/form/{id}`
- Partially update a form `PATCH /api/rest/2.0/assets/form/{id}`

The following processing step types are impacted:

- `"type": "FormStepCreateUpdateContactFromFormField"`
- `"type": "FormStepCreateUpdateAccountFromFormField"`
- `"type": "FormStepCreateUpdateCustomObjectFromFormField"`

See an example

Pre 23A

Request

```
GET /api/rest/2.0/assets/form/12345
```

Response

```
{
"type": "Form",
    "currentStatus": "Draft",
    "id": "12345",...
    "elements": [
        {
        "type": "FormField",
        "id": "2584",
```

```
        "name": "Email",
        ...
        },
        {
        "type": "FormField",
        "id": "2583",
        "name": "City",
              ...
        },
        {
        "type": "FormField",
        "id": "2592",
        "name": "Favorite Colour",
        ...
        },
        {
        "type": "FormField",
        "id": "2587",
        "name": "Submit",
        ...
        }
    ],...
    "processingSteps": [
        {
        "type": "FormStepCreateUpdateContactFromFormField",
        "id": "1826",
        "name": "Create / Update Contact, Prospect or Company",
        "execute": "always",
```

```
            "keyFieldId": "100001",
            "mappings": [
                {
                "type": "FormFieldUpdateMapping",
                "sourceFormFieldId": "2583",
                "targetEntityFieldId": "100009",
                "updateType": "useFieldRule"
                },
                {
                "type": "FormFieldUpdateMapping",
                "sourceFormFieldId": "2584",
                "targetEntityFieldId": "100001"
                }
            ]
            }
    ],...
}
```

Post 23A

Request

```
GET /api/rest/2.0/assets/form/12345
```

Response

```
{
"type": "Form",
```

```
"currentStatus": "Draft",
"id": "12345",...
"elements": [
      {
      "type": "FormField",
      "id": "2584",
      "name": "Email",
      ...
      },
       {
      "type": "FormField",
      "id": "2583",
      "name": "City",
            ...
      },
      {
      "type": "FormField",
      "id": "2592",
      "name": "Favorite Colour",
      ...
      },
      {
      "type": "FormField",
      "id": "2587",
      "name": "Submit",
      ...
       }
],...
```

```
    "processingSteps": [
        {
        "type": "FormStepCreateUpdateContactFromFormField",
        "id": "1826",
        "name": "Create / Update Contact, Prospect or Company",
        "execute": "always",
        "keyFieldId": "100001",
        "mappings": [
            {
            "type": "FormFieldUpdateMapping",
            "sourceFormFieldId": "2583",
            "targetEntityFieldId": "100009",
            "updateType": "useFieldRule"
            },
            {
                "type": "FormFieldUpdateMapping",
                "sourceFormFieldId": "2592",
            },
            {
            "type": "FormFieldUpdateMapping",
            "sourceFormFieldId": "2584",
            "targetEntityFieldId": "100001"
            }
        ]
        }
    ],...
}
```

- 409 Concurrent Form Update Error: Multiple concurrent updates for forms using the Oracle Eloqua UI or the Application APIs 1.0 and 2.0 endpoints will now be prevented. A `409 Conflict` response will be returned via API, and in the UI a warning box will appear. For more information, see the product notice.

  The following endpoints now return a `409 Conflict` response:

  - Update a form version 1.0 `PUT /api/REST/1.0/assets/form/{id}`
  - Update a form version 2.0 `PUT /api/rest/2.0/assets/form/{id}`
  - Partially update a form `PATCH /api/rest/2.0/assets/form/{id}`

  See an example

  Example 409 conflict response:

  Request (more than one at the same time to this endpoint):

  ```
  PUT /api/rest/2.0/assets/form/12345
  ```

  Response

  ```
  Header: 409 CONFLICT
  Body:
  [{"type":"ConcurrentUpdateError","details":"Your save was not
  completed due to a recent modification of this form by another
  user."}]
  ```

Release 22D

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 22D.

New features

Bulk API

- Bulk API contact app exports can now include source Campaign ID and fields. When a contact is moved from a Campaign Canvas to a Program Canvas the source Campaign Canvas will now be tracked. The Source Campaign ID and fields can then be included in a Bulk API Contact app export within the Program Canvas. Include the Campaign ID using the following statement: `{{Contact.SourceCampaign.Id}}`. Add the source Campaign fields using this syntax: `{{Contact.SourceCampaign.Field(<FieldName>)}}`. If a `SourceCampaign` field is included in an export definition that is not for a Program Canvas app, the export definition creation will fail.

- All Special Fields (e.g. Id) are now included in Fields endpoints.
  - `{{Contact.Email.IsSubscribed}}`
  - `{{Contact.Email.IsBounced}}`
  - `{{Contact.Email.Format}}`
  - `{{Contact.Id}}`
  - `{{Account.Id}}`
  - `{{CustomObject[<customObjectId>].Id}}`
  - `{{Event[<eventId>].Id}}`

Platform notices

- With the arrival of Eloqua 22D release (November 2022), the EloquaService SOAP API will be deleted. For more information, see the product notice.

Release 22C

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 22C.

New features

Application API

- We have added a new property `isUpdatingCrmID` to the **Update a campaign** endpoint. This property indicates if `crmId` is being updated. For more information, see the product notice.

Bulk API

- We have added new opportunity fields `Owner` and `PrimaryCampaignID` for Bulk API import. You can retrieve these new opportunity fields using the API endpoint:

  `GET/API/Bulk/2.0/opportunities/fields`.

  See the API reference for more information.

  See an example

  HTTP request

  ```
  POST /opportunities/imports
                                  Content-Type: application/json
  ```

  Request body

```
{
                                    "name": "Opportunity Import",
                                    "fields": {
                                    "OpportunityID": "{{Opportunity.Id}}",
                                    "OpportunityName": "{{Opportunity.Field(Name)}}",
                                    "AccountName": "{{Opportunity.Field(AccountName)}}",
                                    "CreatedDate": "{{Opportunity.CreatedAt}}",
                                    "Amount": "{{Opportunity.Field(Amount)}}",
                                    "CloseDate": "{{Opportunity.Field(CloseDate)}}",
                                    "Currency": "{{Opportunity.Field(Currency)}}",
                                    "ForecastToCloseDate": "{{Opportunity.Field(ForecastToCloseDate)
                                    "Stage": "{{Opportunity.Field(Stage)}}",
                                    "Territory": "{{Opportunity.Field(Territory)}}",
                                    "Owner": "{{Opportunity.Field(Owner)}}",
                                    "PrimaryCampaignId": "{{Opportunity.Field(PrimaryCampaignId)}}"
                                    },
                                    "identifierFieldName": "OpportunityID",
                                    "isIdentifierFieldCaseSensitive": false
                                }
```


Response


```
{
                                    "isIdentifierFieldCaseSensitive": false,
                                    "name": "Opportunity Import",
                                    "fields": {
                                    "OpportunityID": "{{Opportunity.Id}}",
                                    "OpportunityName": "{{Opportunity.Field(Name)}}",
                                    "AccountName": "{{Opportunity.Field(AccountName)}}",
```

```
                "CreatedDate": "{{Opportunity.CreatedAt}}",
                "Amount": "{{Opportunity.Field(Amount)}}",
                "CloseDate": "{{Opportunity.Field(CloseDate)}}",
                "Currency": "{{Opportunity.Field(Currency)}}",
                "ForecastToCloseDate": "{{Opportunity.Field(ForecastToCloseDate)
                "Stage": "{{Opportunity.Field(Stage)}}",
                "Territory": "{{Opportunity.Field(Territory)}}",
                "Owner": "{{Opportunity.Field(Owner)}}",
                "PrimaryCampaignId": "{{Opportunity.Field(PrimaryCampaignId)}}"
                },
                "identifierFieldName": "OpportunityID",
                "isSyncTriggeredOnImport": false,
                "dataRetentionDuration": "P7D",
                "uri": "/opportunities/imports/6769",
                "createdBy": "API.User",
                "createdAt": "2022-06-13T14:02:29.6270000Z",
                "updatedBy": "API.User",
                "updatedAt": "2022-06-13T14:02:29.6270000Z"
        }
```


Recent changes

Application API

- We have added a new search parameter `isCertificateProvisioned` to the Retrieve a list of microsites endpoint. Set `isCertificateProvisioned` to “true” to filter on microsites that have a certificate provisioned for at least one of your domains. If `isCertificateProvisioned` is not included in request all microsites will be returned.

Platform notices

- With the arrival of Eloqua 22D release (November 2022), the EloquaService SOAP API will be deleted. For more information, see the product notice.

Release 22B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 22B.

New features

Application API

- We've added a new query parameter that enables you to retrieve a list of Microsites provisioned using the Automated Certificate Management system.

  See an example

  Retrieve a list of Microsites provisioned using the Automated Certificate Management system:

  ```
  GET /api/REST/1.0/assets/microsites?isCertificateProvisioned=true
  ```

  For more information, see the API reference.

Bulk API

- We've added the `updateRuleByField` parameter to Bulk API imports. This parameter specifies how Eloqua should handle updating records when you import a new value for an existing field by field. If you do not specify an `updateRuleByField`, or leave a field out of `updateRuleByField`, the rule type defaults to the rule set with `updateRule`. The following

rule types are available:

- `always`: Always update.
- `ifNewIsNotNull`: Update if the new value is not blank.
- `ifExistingIsNull`: Update if the existing value is blank.
- `useFieldRule`: Use the rule defined at the field level.

See an example

Create a new contact import definition and set the update rule by field, for some fields.

HTTP request

```
POST /api/bulk/2.0/accounts/imports
                                    Content-Type: application/json
```

Request body

```
{
                                        "name": "updateRuleByField Import",
                                        "fields": {
                                        "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
                                        "FirstName": "{{Contact.Field(C_FirstName)}}",
                                        "LastName": "{{Contact.Field(C_LastName)}}",
                                        "Company": "{{Contact.Field(C_Company)}}",
                                        "Address1": "{{Contact.Field(C_Address1)}}",
                                        "Address2": "{{Contact.Field(C_Address2)}}",
                                        "City": "{{Contact.Field(C_City)}}",
                                        "StateProvince": "{{Contact.Field(C_State_Prov)}}"
                                        },
```

```
                "identifierFieldName": "EmailAddress",
                "updateRule": "always",
                "updateRuleByField": {
                "FirstName": "ifNewIsNotNull",
                "LastName": "useFieldRule",
                "Company": "ifExistingIsNull"
                }
            }
```

Response

```
{
                "name": "updateRuleByField Import",
                "updateRule": "always",
                "fields": {
                "EmailAddress": "{{Contact.Field(C_EmailAddress)}}",
                "FirstName": "{{Contact.Field(C_FirstName)}}",
                "LastName": "{{Contact.Field(C_LastName)}}",
                "Company": "{{Contact.Field(C_Company)}}",
                "Address1": "{{Contact.Field(C_Address1)}}",
                "Address2": "{{Contact.Field(C_Address2)}}",
                "City": "{{Contact.Field(C_City)}}",
                "StateProvince": "{{Contact.Field(C_State_Prov)}}"
                },
                "updateRuleByField": {
                "FirstName": "ifNewIsNotNull",
                "LastName": "useFieldRule",
                "Company": "ifExistingIsNull"
                },
```

```
            "identifierFieldName": "EmailAddress",
            "isSyncTriggeredOnImport": false,
            "dataRetentionDuration": "P7D",
            "isUpdatingMultipleMatchedRecords": false,
            "uri": "/contacts/imports/854",
            "createdBy": "API.User",
            "createdAt": "2022-02-24T23:48:28.2330000Z",
            "updatedBy": "API.User",
            "updatedAt": "2022-02-24T23:48:28.2330000Z"
        }
```


Learn more.

Recent changes

- The new Contact Field setting "Contact Field contains valid phone number" prevents updating contact information if the contact record does not contain a valid phone number, using both Application API and Bulk API endpoints. Endpoints to update contact records are affected such as the Update a contact endpoint and Update a contact import definition. When using the Bulk API, clients with the "Reject rows that have invalid data" feature enabled will also notice that contacts will be rejected if they contain invalid phone numbers.

Application API

- If there are over one million CDO (custom data object) records, the Retrieve a list of custom object data API 1.0 and 2.0 endpoints will now only accept one search request at a time if the request includes only non-indexed fields or does not include an indexed field without a leading wildcard. This limit will be per User and per Custom Object. A “429 Too Many Requests” response will be returned until the request completes.

  For more information, see the product notice.

Bulk API

- We have modified how Contacts are linked to Bulk API form submit activities. Eloqua will now check first for the Contact ID that was set upon form submission, and if there is not a Contact ID set upon form submission, Eloqua will then check for a Contact that's mapped to the Visitor.

  For more information, see the product notice.

Platform notices

- With the arrival of Eloqua 22D release (November 2022), the EloquaService SOAP API will be deleted. For more information, see the product notice.

Documentation enhancements of note

- The Mapping contacts via form submit now includes more Considerations.
- We've updated the Determining base URLs tutorial to include the new pod 8.

Product notices

- We have modified the Eloqua Application API endpoints used to retrieve users so that minimal depth will not return `createdBy` and `updatedBy`. For more information, see the product notice.

Release 22A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 22A.

New features

- We've added a new property to the Form 2.0 Application API endpoints when Form Spam Protection is enabled (`isFormSpamProtectionEnabled`). Learn more.

Recent changes

Bulk API

- The sync logs endpoint now groups Eloqua status codes ELQ-00111 and ELQ-00121 into one message, instead of displaying multiple messages for each missing or ignored field. This enhancement reduces the number of messages reported within sync logs.

  See an example

  Sync logs before 22A:


```
{
                                    "items": [
                                    {
                                    "syncUri": "/syncs/2865",
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00121",
                                    "message": "Field (C_FirstName) is part of import definition, but no
file .",
                                    "createdAt": "2021-10-27T23:36:34.8930000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/2865",
                                    "count": 0,
                                    "severity": "warning",
```

```
"statusCode": "ELQ-00121",
"message": "Field (C_LastName) is part of import definition, but not
file .",
"createdAt": "2021-10-27T23:36:34.8930000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "warning",
"statusCode": "ELQ-00111",
"message": "Field (notindeffield1) is not part of import definition, an
ignored.",
"createdAt": "2021-10-27T23:36:34.8930000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "warning",
"statusCode": "ELQ-00111",
"message": "Field (notindeffield2) is not part of import definition, an
ignored.",
"createdAt": "2021-10-27T23:36:34.8930000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "warning",
"statusCode": "ELQ-00111",
```

```
                                "message": "Field (notindeffield3) is not part of import definition, a
ignored.",
                                "createdAt": "2021-10-27T23:36:34.8930000Z"
                                },
                                {
                                "syncUri": "/syncs/2865",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield4) is not part of import definition, a
ignored.",
                                "createdAt": "2021-10-27T23:36:34.8930000Z"
                                },
                                {
                                "syncUri": "/syncs/2865",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield5) is not part of import definition, a
ignored.",
                                "createdAt": "2021-10-27T23:36:34.8930000Z"
                                },
                                {
                                "syncUri": "/syncs/2865",
                                "count": 1,
                                "severity": "information",
                                "statusCode": "ELQ-00130",
                                "message": "Total records staged for import.",
```

```
"createdAt": "2021-10-27T23:36:34.9870000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00137",
"message": "Ready for data import processing.",
"createdAt": "2021-10-27T23:36:35.0200000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00101",
"message": "Sync processed for sync , resulting in Warning status."
"createdAt": "2021-10-27T23:36:35.7800000Z"
},
{
"syncUri": "/syncs/2865",
"count": 1,
"severity": "information",
"statusCode": "ELQ-00001",
"message": "Total records processed.",
"createdAt": "2021-10-27T23:36:34.6200000Z"
},
{
"syncUri": "/syncs/2865",
```

```
                "count": 1,
                "severity": "information",
                "statusCode": "ELQ-00004",
                "message": "Contacts created.",
                "createdAt": "2021-10-27T23:36:35.2770000Z"
                },
                {
                "syncUri": "/syncs/2865",
                "count": 0,
                "severity": "information",
                "statusCode": "ELQ-00022",
                "message": "Contacts updated.",
                "createdAt": "2021-10-27T23:36:35.2770000Z"
                }
                ],
                "totalResults": 13,
                "limit": 1000,
                "offset": 0,
                "count": 13,
                "hasMore": false
            }
```


Sync logs in 22A:


```
{
                "items": [
                {
                "syncUri": "/syncs/2865",
```

```
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00121",
                                    "message": "Field (C_FirstName,C_LastName) is part of import defi
included in file .",
                                    "createdAt": "2021-10-27T23:36:34.8930000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/2865",
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00111",
                                    "message": "Field
(notindeffield1,notindeffield2,notindeffield3,notindeffield4,notindeffield5) is not part
of import definition, and will be ignored.",
                                    "createdAt": "2021-10-27T23:36:34.8930000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/2865",
                                    "count": 1,
                                    "severity": "information",
                                    "statusCode": "ELQ-00130",
                                    "message": "Total records staged for import.",
                                    "createdAt": "2021-10-27T23:36:34.9870000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/2865",
                                    "count": 0,
```

```
"severity": "information",
"statusCode": "ELQ-00137",
"message": "Ready for data import processing.",
"createdAt": "2021-10-27T23:36:35.0200000Z"
},
{
"syncUri": "/syncs/2865",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00101",
"message": "Sync processed for sync , resulting in Warning status."
"createdAt": "2021-10-27T23:36:35.7800000Z"
},
{
"syncUri": "/syncs/2865",
"count": 1,
"severity": "information",
"statusCode": "ELQ-00001",
"message": "Total records processed.",
"createdAt": "2021-10-27T23:36:34.6200000Z"
},
{
"syncUri": "/syncs/2865",
"count": 1,
"severity": "information",
"statusCode": "ELQ-00004",
"message": "Contacts created.",
"createdAt": "2021-10-27T23:36:35.2770000Z"
```

```
        },
        {
        "syncUri": "/syncs/2865",
        "count": 0,
        "severity": "information",
        "statusCode": "ELQ-00022",
        "message": "Contacts updated.",
        "createdAt": "2021-10-27T23:36:35.2770000Z"
        }
        ],
        "totalResults": 8,
        "limit": 1000,
        "offset": 0,
        "count": 8,
        "hasMore": false
}
```


- Resolved issue that caused uploaded data to possibly be assigned to a previously created sync and cause other syncs to fail with the error "Import Sync creation aborted, there's no data to import.". This issue was only applicable when uploading data to import definitions created with `isSyncTriggeredOnImport=true`.

Documentation enhancements of note

- We've updated the List of terms supported with search within the Using the search URL parameter tutorial.
- See our new tutorial Using form spam protection with the Application API.

Release 21D

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 21D.

Recent changes

Bulk API

- We've updated the Bulk API limits. Every Eloqua instance has the following soft limits by Bulk API sync type:

Bulk API Sync Type	Limit
Export	2000 per hour
Import	2000 per hour
Sync Action	4000 per hour

  There are many variables to sync processing. We'd expect minimal impact due to volume on its own if within these limits and following best practices.

  Learn more about Bulk limits.

Documentation enhancements of note

- We've added an example to the Using the search URL parameter tutorial, under Operators.

  See the example

  If including more than one different field, they will be ANDs.

```
?search=name='Test1'name='Test2'C_Company='Oracle'
```

This request will return records with a name of "Test1" or "Test2" and a Company of "Oracle".

- The Sync actions page now includes a Sync Action App Status section. This section explains the sync action app statuses, the status description displayed, and when each status is set.
- We've updated the Bulk API limits page to reflect the new Bulk API limits.

Release 21C

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 21C.

New features

Application API

- Asynchronous Form Processing via API is now available for the Create form data for a single form endpoint. Learn more.

Authentication

- More detailed 401 error responses are now available when authenticating with OAuth 2.0:
  - For the https://login.eloqua.com/auth/oauth2/token endpoint, for the error code "401 Unauthorized", the response body will now include an error, error code, and error description. For example:

```
{
                                        "error": "unknown_token",
                                        "error_code": 2501,
                                        "error_description": "Provided Token is Unknown."
```

```
}
```

Here's a description of each property:

- `error`: The name of the error type.
- `error_code`: The numeric value uniquely identifying the type of error.
- `error_description`: Provides details on the error.

- Errors have been divided into categories indicated by the 1000s digit of the `error_code` according to the following:
  - `1000`: General Error Messages
  - `2000`: Error Messages Related to Authentication
  - `2500`: Error Messages Related to OAuth2.0 Authentication
  - `3000`: Error Messages Related to Authorization

For more information see the product notice, or refer to the documentation.

Bulk API

- **Update August 27th 2021**: This feature is no longer included in 21C.

  ~~Bulk API imports now accept json encoded in surrogate pairs, e.g. `\uD83D\uDE03`.~~

- When an external ID is used as the `identifierFieldName` for imports, the Bulk API will now extract the ID and match on the ID. This change improves the performance of imports. The following external IDs are extracted and matched:

```
{{Contact.Field(ContactIDExt)}}
                    {{Account.Field(CompanyIDExt)}}
                    {{CustomObject[<id>].ExternalId}}
                {{Event[<id>].ExternalId}}
```

Recent changes

App Developer Framework

- When there is no response to the Notification URL call from the App, Eloqua will now retry the Notification URL call over approximately an eight-hour period of time instead of failing after one attempt
- For Content services, Eloqua will now set records to "error" as soon as errors occur. Previously there was a 24 hour delay.

Bulk API

- Resolved an issue where the `createdAt` timestamp in the Create sync endpoint response was not matching the `createdAt` timestamp in the Retrieve a sync endpoint response.
- We have added an improved retry strategy to cover a wider variety of failures that can occur during Bulk API export.

Platform notices

- Oracle will be modifying its supported cipher suites used for Transport Layer Security (TLS) connections to Eloqua. This includes programmatic access to Eloqua via APIs. Additional Ciphers were added on August 1, 2021 and weak Ciphers will be removed on October 1, 2021, providing customers with a 2 month window for testing in between. For more information see the product notice.

Documentation enhancements of note

- We've added a new tutorial that explains how using the Multiple Branded Domains feature changes the way you interact with the Application API endpoints. See Using multiple branded domains with the application API for more information.

Release 21B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 21B.

New features

Application API

- The default depth for the Application API Retrieve batch form submission details endpoints has been changed to minimal (previously, the default depth was complete). For more information, see the product notice.

App Developer Framework

- `CampaignId` has been added as a URL template parameter to the Content Service Landing Page Notification URL. The `CampaignId` will be populated with the `elqCampaignId` query parameter.

Recent changes

Application API

- Updating a value for custom field which is set to `uniqueCode` on Custom Object will now assign value to `uniqueCode` for the Custom Object record.
- A validation error has been added for the Custom Object endpoints, so that an error is returned if the same field is included more than once.

App Developer Framework

- The following new app member status has been added "Import Queued by App". This message is displayed when an App has created an import to update a step member's status.

See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

Bulk API

- Emails sent from the Outlook Plugin are now displayed as `OutlookPlugin` for the EmailSend Type in Bulk API activity exports.
- Bulk API dependencies now appear when there is a Bulk API event definition.
- We have added an improved retry strategy to cover a wider variety of failures that can occur during Bulk API export.

Platform notices

- With the arrival of Eloqua Release 21C, Asynchronous Form Processing via API will be released for the Create form data for a single form endpoints. This feature can be enabled for Early Preview with Eloqua Release 21B.

Release 21A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 21A.

New features

Application API

- We've added a new endpoint that enables you to retrieve spam form submissions. A successful response will return a list of all spam form submissions for the form. This endpoint is released under our Controlled Availability program. Please contact your account representative for more information. For more information about the endpoint, see the API

Reference.

See an example

Retrieve form submissions for the form with ID 3 where the form submissions were blocked because of an invalid token:

```
GET
/api/REST/2.0/data/forms/3/formSpamData?search=blockreason=invalidtoken
```

Response:

```
{
                                    "elements": [
                                    {
                                    "type": "FormSpamData",
                                    "id": "1236",
                                    "blockReason": "invalidtoken",
                                    "fieldValues": [
                                    {
                                    "type": "FieldValue",
                                    "name": "firstName",
                                    "value": "John"
                                    },
                                    {
                                    "type": "FieldValue",
                                    "name": "lastName",
                                    "value": "Smith"
                                    },
                                    {
```

```
            "type": "FieldValue",
            "name": "emailAddress",
            "value": "john.smith@oracle.com"
            },
            {
            "type": "FieldValue",
            "name": "elqFormName",
            "value": "UntitledForm-1536934056562"
            },
            {
            "type": "FieldValue",
            "name": "elqSiteId",
            "value": "426809016"
            }
            ],
            "formSubmissionSource": "get",
            "submittedAt": "1610023989"
            }
            ],
            "page": 1,
            "pageSize": 1000,
            "total": 1
        }
```


Bulk API

- We've added a new campaign response field `{{CampaignResponse.AddedAt}}`. This new field enables you to determine the date and time campaign responses were added to your Eloqua database, when creating campaign response export definitions. The Bulk API

Campaign Response export will now use `AddedAt` instead of `CreatedAt` when enforcing the 30 day limit of campaign responses you can retrieve.

Recent changes

Application API

- Validation has been added to the Application API endpoint to Update a contact field to disallow changing a contact field from Text box, Check box, or Single-select list to a Multi-select list.

  See an example of the new validation message


```
[
                                        {
                                        "type": "ObjectValidationError",
                                        "container": {
                                        "type": "ObjectKey",
                                        "objectId": "{ID}",
                                        "objectType": "ContactField"
                                        },
                                        "property": "displayType",
                                        "requirement": {
                                        "type": "PreconditionRequiredRequirement",
                                        "description": "Display type cannot be set to multi-select
fields that are not a multi-select picklist already."
                                        },
                                        "value": "multiSelect"
                                        }
                                    ]
```

- The status message "Eloqua configuration error – Manage Data Export Permission" has been updated to allow the app to set this status. App developers can set this status for Action and Decision services using sync actions or export filters. Previously this could only be set by Eloqua if records were set to be returned in the Notify call. See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

  See an example of setting the new permission status with a sync action

```
{
                                        "action": "setStatus",
                                        "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
                                        "status": "permission",
                                        "updateAll": true
                                    }
```

- We have improved the App Developer Notification call, by adding a retry strategy to cover a wider variety of failures that can occur with sending notification calls.

Documentation enhancements of note

- We've added new notes for multiple Eloqua status codes to better help you troubleshoot.

Release 20D

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 20D.

New features

Application API

- Batch processing of forms via API is now generally available. This allows you to create API Form Submission payloads in batches of 500. For more information, see the API Documentation.

- We've made the following enhancements to the Email endpoints to support email archiving:

    - There are two new API endpoints to enable archiving and unarchiving emails. For more information, see the API Documentation.

      See an example

      Archive an email with the email ID 5 into the folder with the folder ID 10:

      ```
      POST/api/REST/2.0/assets/email/5/archive
                                                  {
                                                  "targetFolderId": "10"
                                              }
      ```

    - The Update an email endpoints in 1.0 and 2.0 has new validation to prevent updating archived emails.

    - The Retrieve a list of emails endpoints in the Application API 1.0 and 2.0 now only returns unarchived emails by default.

    - Two new query parameters are available with the Retrieve a list of emails endpoint in the Application 2.0 endpoint: `includeAvailable` and `includeArchived`. These new query parameters enable you to filter archived emails. For more information, see the API Documentation.

      See an example

      Retrieve archived emails:

```
GET
/api/REST/2.0/assets/emails?includeAvailable=false&includeArch
ived=true
```

- The `archive` property has been renamed to `archived`. For more information, see the product notice.

- We've added support for the new type `EmailInlineDeployment` with the Email Deployment endpoint to enable sending HTML emails, instead of having to specify an existing Eloqua email, as well as being able to schedule email deployments using the `sendDate` parameter. Emails can be sent to up to 2000 contacts per request.

  For more information, see the API Documentation.

  See an example

  Create and send an email deployment for an existing email asset to two contacts on Thursday, January 20, 2022.

```
POST/assets/email/deployment
                                {
                                "type": "EmailInlineDeployment",
                                "depth": "complete",
                                "name": "Email Send Future Date",
                                "email": {
                                "type": "Email",
                                "name": "Test Inline Deployment",
                                "id": 101,
                                "htmlContent": {
                                "type": "RawHtmlContent",
```

```
                                    "html": "<html><head></head><body>Test Inline
Deployment</body></html>"
                                    }
                                    },
                                    "sendDate": "1642703971",
                                    "sendFromUserId": 1,
                                    "contacts": [
                                    {
                                    "id": 2,
                                    "id": 4
                                    }
                                    ]
                                }
```

- A new property, `crmSystemMappings`, is now returned at `complete` depth with the Retrieve an account Application API endpoint. This property is returned when the Account is linked to an Eloqua User, and indicates the CRM User Names and Eloqua Login Name for each Eloqua User linked to the Account. For more information, see the product notice.

  See an example

  Retrieve an account at complete depth:

```
GET/API/REST/1.0/data/account/8?depth=complete
                                    {
                                    "type": "Account",
                                    "id": "1",
                                    "createdAt": "1423758306",
                                    "depth": "complete",
```

```
                "description": "This is an example company.",
                "name": "Cyberdyne Systems",
                "updatedAt": "1423758306",
                "address1": "123 Industry Lane",
                "address2": "",
                "address3": "",
                "businessPhone": "5555555555",
                "city": "Los Angeles ",
                "country": "United States",
                "crmSystemMappings": [
                {
                "type": "CrmSystemMapping",
                "OSCUserName": "12345678",
                "loginName": "CRM User"
                }
                ],
                "fieldValues": [],
                "postalCode": "",
                "province": "CA"
            }
```


Bulk API

- You can now use the Bulk API Account Import to link Eloqua Users to Accounts. By specifying a CRM User Id, you can create a link between Eloqua Users and Accounts to be used by integrations.

  See an example

  Create an Account Import definition, and link users using the field "Oracle Sales Cloud User ID" (CRMOSCUserId):

```
POST /accounts/imports
                                        {
                                        "name": "Import Accounts and link to Users",
                                        "fields": {
                                        "CompanyName": "{{Account.Field(M_CompanyName)}}
                                        "UserCRMIdLink": "{{Account.User.Field(CRMOSCUserId
                                        },
                                        "identifierFieldName": "CompanyName"
                                        "linkUsers": "true"
                                    }
```

- When creating a Bulk API sync, the definition will now be validated. We've also added a new sync log message to inform you if syncs fail because of a temporary error. For more information, see the product notice.

  See an example

```
{
                                        "syncUri": "/syncs/2651",
                                        "count": 0,
                                        "severity": "information",
                                        "statusCode": "ELQ-00146",
                                        "message": " Unable to process sync due to a temporary
                                        "createdAt": "2020-09-01T22:45:55.1570000Z"
                                    }
```

Recent changes

App Developer Framework

- We've added two new app member statuses, one surfaces when the Eloqua User does not have access to Export records, and the other displays when the app step is waiting for the scheduled send period. Additionally, the "Invalid app configuration" status has been updated to include when the export fails because of an invalid definition. See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

Application API

- A new URL query parameter, `ownedByUserID`, is now available for the Retrieve an account endpoint endpoint. Set this query parameter to an Eloqua User Id to return all Accounts linked to that Eloqua User. Example: This request will return all Accounts linked to Eloqua User Id 71:

```
GET /api/REST/1.0/data/accounts?ownedByUserID=71
```

Bulk API

- A new retention policy is now in effect for Bulk API syncs including sync logs and rejects (older than one year). This new retention policy will continue to be applied from 20D on. For more information, see the product notice.

Documentation enhancements of note

- We've added a new topic that details how to import account data.

Product Notices

- Oracle modified its supported cipher suites used for Transport Layer Security (TLS) connections to Eloqua. This includes programmatic access to Eloqua via APIs. This change

has been rolled back. For more information, see the product notice.

Release 20C

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 20C.

New features

- You can now write to the "Last Modified by CRM" system field for Contacts and Accounts. This enhancement enables developers of CRM integrations to indicate when a Contact and Account has been updated by CRM.

Application API

- We've added new Application 2.0 API endpoints that enable you to manage Users. Along with managing Users, these new endpoints enable you to perform actions that are not available in 1.0 such as Enable or Disable a User and Update a User's Password. For more information, see the API Documentation.
- We've added new API endpoints that enable you to manage Eloqua Security Groups. For more information, see the API documentation.

Recent changes

App Developer Framework

- We've updated our retry strategy when Eloqua calls an App's Notification URL. Eloqua will now retry the Notification URL call over approximately an eight-hour period of time with a backoff strategy. See the product notice to learn more.
- We've implemented new detailed status messaging for stuck records. The following new app member statuses have been added:

- Awaiting app to process: App Service Settings Step Response is set to None
- Authentication required - Reinstall app: App does not have a valid token
- Invalid app configuration: App sets status to invalid
- Eloqua notifying app: Eloqua is calling the app's Notification URL
- Being processed by app: This status used to be Being processed by external service
- Eloqua failed to notify app: All of Eloqua's calls to app's Notification URL have failed. Will be routed to error path.
- Status Error set by Eloqua: There was an Eloqua error that was not related to an invalid definition. Will be routed to error path.
- App notified successfully - Awaiting app to process: Eloqua received 200 level response to the Notification URL from the app (when Service Setting of Records per Notification is set to 0)
- Status 'Error' set by app: App sets status to error. Will be routed to error path.

- We've added new validation to ensure app service instances exist, and in cases where service instances do not exist, a validation error will be returned. This validation applies when creating or updating import and export definitions, creating syncs, and executing syncs.
  - See an example

    Here's an example where a validation error is returned when attempting to create or update a definition referencing an app service instance ID that does not exist.

```
{
                                                    "failures": [
                                                    {
                                                    "field": "filter",
                                                    "value": "STATUS('{{ActionInstance
(2a9f17e8ca374447a8e6ceb24ccd246)}}') = 'pending'",
                                                    "constraint": "Must be a reference to an e
                                                    }
```

```
                                    ]
                                }
```

Application API

- The Contact Field endpoints no longer return the `isPopulatedInOutlookPlugin` property. See the product notice for more information.

Platform notices

- With the arrival of Eloqua release 20D (Nov 2020), we will begin applying a new retention policy on Bulk API syncs, including sync logs and rejects, that are older than one year. This new retention policy will continue to be applied from 20D onwards. For more information, see the product notice.

Documentation enhancements of note

- We've added a new topic that explains app statuses. See the Oracle Eloqua Help Center for a list of all the different status messages you can see related to app members moving through your campaigns and programs.

Release 20B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 20B.

New features

Application API

- We've added new API endpoints to enable you to archive and unarchive forms programmatically. Archiving forms enables you to hide older forms from search, choosers,

and newer marketing activities. Forms in archive can still accept form submissions, but cannot be edited, copied or used in new campaigns or content.

A new property has been added to the Forms API endpoints in 1.0 and 2.0. The `archived` property is a boolean which indicates whether or not a form is in archive.

The Update a form endpoints will have new validation to prevent updating archived forms, and we've also made changes to the Retrieve a list of forms endpoints. Retrieve a list of forms endpoint will only return unarchived forms by default. Two new URL parameters will be available with the Retrieve a list of forms endpoint: `includeAvailable` and `includeArchived`, which will enable filtering results on the archived state.

For more information, see the product notice and API documentation.

- Archive Form Example

  Archive a form with the form ID 2.

  ```
  POST /api/REST/2.0/assets/forms/2/archive
  ```

  Move the form into a specific folder.

  ```
  {
                                                  "targetFolderId": "5"
                                              }
  ```

  Response body:

  ```
  {
                                                  "type": "Form",
                                                  "currentStatus": "Draft",
                                                  "id": "2",
                                                  "createdAt": "1418404077",
  ```

```
                "createdBy": "5",
                "depth": "minimal",
                "folderId": "5",
                "name": "Form 01",
                "permissions": [
                "Retrieve",
                "SetSecurity",
                "Delete",
                "Update"
                ],
                "updatedAt": "1534172012",
                "updatedBy": "9",
                "archived": "true",
                "htmlName": "untitledForm-14184040758
                "isResponsive": "false"
        }
```


- Unarchive Form Example

  Unarchive a form with the ID 2.

  ```
  POST /api/REST/2.0/assets/forms/2/unarchive
  ```

  Move the form into a specific folder.


  ```
  {
                "targetFolderId": "6"
        }
  ```


  Response body:

```
{
                                        "type": "Form",
                                        "currentStatus": "Draft",
                                        "id": "2",
                                        "createdAt": "1418404077",
                                        "createdBy": "5",
                                        "depth": "minimal",
                                        "folderId": "6",
                                        "name": "Form 01",
                                        "permissions": [
                                        "Retrieve",
                                        "SetSecurity",
                                        "Delete",
                                        "Update"
                                        ],
                                        "updatedAt": "1534172012",
                                        "updatedBy": "9",
                                        "archived": "false",
                                        "htmlName": "untitledForm-14184040758
                                        "isResponsive": "false"
                                    }
```


Bulk API

- We've added a new activity field named `FormSubmitSavedId`. This field is available when creating an activity export definition for form submit activity types. This new activity field enables you to include the identifier of when form submissions were saved in your form submit activity reports, so you can export form submit activities periodically without missing any.

  The syntax is:

- FormSubmitSavedId: `{{Activity.Field(FormSubmitSavedId)}}`

When export data is retrieved, the `FormSubmitSavedId` will be populated with the unique identifier for when the form submission was saved.

See an example

Here's sample export data that includes `FormSubmitSavedId`:

```
{

"firstName=Text&lastName=&annualRevenue1=&emailAddre
ss=&country=&leadStatus1=&leadScoreExplicit1=&salesp
erson=&multipleSelectMenu=&submit=",

}
```

Recent changes

App Developer Framework

- Resolved an issue related to receiving content from an app to improve consistency of content services.

Authentication

- If the current access token has not been used, submitting a request to authenticate with the previous refresh token will now return the existing new access token and refresh token. This update alleviates the need to have to reinstall an app if there is a server error during token refresh.

Bulk API

- Resolved an issue where it was possible to create custom object export definitions that included a contact lead scoring field, even though contact lead scoring fields are not supported in custom object exports, and the sync would always fail. Now, creating a custom object export definition that includes a contact lead scoring field will result in a 400 validation error.

  Example

  Create a custom object export definition with the contact lead scoring field:

```
POST /API/Bulk/2.0/customObjects/9/exports
                                        {
                                        "name": "CustomObjectExportwithLeadScoringField",
                                        "fields": {
                                        "coEmail": "{{CustomObject[9].Field[103]}}",
                                        "coId": "{{CustomObject[9].ExternalId}}",
                                        "C_FirstName": "{{CustomObject[9].Contact.Field(C_First
                                        "contactID": "{{CustomObject[9].Contact.Id}}",
                                        "contactLeadScoreRating": "{{CustomObject
[9].Contact.LeadScore.Model[1].Rating}}"
                                        }
                                    }
```

  Response:

400 There was a validation error.


```
{
"failures": [
{
"field": "contactLeadScoreRating",
"stackTrace": [
{
"field": "fields"
}
],
"value": "{{CustomObject[9].Contact.LeadScore.Model[1]
"constraint": "LeadScore fields can only be used in conta
}
]
}
```


- We've added a new, more detailed error message for 500 error responses to Bulk API requests. The new error response indicates the type of error, error severity, and when users should retry the request.
    - `error`: Describes the type of error.
    - `errorSeverity`: The severity of the error ranging from 3, 2, 1, and 0. 0 being the highest.
    - `Retry-After`: Based on the error severity, indicates in seconds when the request should be retried.

  This new error message provides Bulk API users with more information regarding failed requests, and guidance on when to retry requests.

  For more information, see the product notice.

  See an example

```
500 The service has encountered an error.
                                        Retry-After: 120
                                        {
                                        "error": "Unable to process request due to a temporary e
                                        "errorSeverity": 3
                                    }
```

- In certain cases where an export sync attempt fails but Eloqua will retry, Eloqua will now return a message indicating Eloqua will retry the sync. The new status code ELQ-00145 indicates your sync will be retried, and the `message` indicates when Eloqua will retry the sync. For more information, see the product notice.

  See an example

```
{
                                        "syncUri": "/syncs/2651",
                                        "count": 0,
                                        "severity": "information",
                                        "statusCode": "ELQ-00145",
                                        "message": "Sync attempt failed. The sync will be retried
seconds.",
                                        "createdAt": "2020-03-01T22:45:55.1570000Z"
                                    }
```

Platform notices

- With the arrival of Eloqua release 20D (Nov 2020), we will begin applying a new retention policy on Bulk API syncs, including sync logs and rejects, that are older than one year. This new retention policy will continue to be applied from 20D onwards. For more information, see the product notice.

Documentation enhancements of note

- Added a list of search terms per endpoint to the Using the search URL parameter tutorial, to provide guidance on which terms to use when searching using the Application API.

Release 20A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 20A.

New features

Application API

- `accountId` will now be returned at `complete` depth when retrieving, creating, or updating Contacts using the Application API when the Contact is linked to an Account. For more information, see the product notice.

Bulk API

- You can now set the status for all records in an execution using just the execution ID. The new `updateAll` parameter determines what status to set records in an execution, when creating a contact or custom object sync action definition.
  Example

  1. Create import definition to set all records in an execution to a status:

     ```
     POST /api/bulk/2.0/contacts/syncActions
     ```

     Request body:

     ```
     {
                                                    "name": "Bulk Sync Action - Action - Set a
     ```

```
                                                    "syncActions": [
                                                    {
                                                    "action": "setStatus",
                                                    "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
                                                    "status": "errored",
                                                    "updateAll": true
                                                    }
                                                    ]
                                            }
```


Response:


```
{
                                                    "name": "Bulk Sync Action - Action - Set a
                                                    "syncActions": [
                                                    {
                                                    "action": "setStatus",
                                                    "destination": "{{ActionInstance
(f82d50cd86a94fcab37e4ec9a98b0339).Execution[12345]}}",
                                                    "status": "errored"
                                                    "updateAll": true
                                                    }
                                                    ],
                                                    "uri": "/contacts/syncActions/635",
                                                    "createdBy": "API.User",
                                                    "createdAt": "2016-01-13T21:32:08.451693
                                                    "updatedBy": "API.User",
```

```
                                                    "updatedAt": "2016-01-13T21:32:08.451693
                }
```

2. Sync the definition to set the status.

Recent changes

Application API

- As part of the Classic Design Editor changes, the Create a form Application API Form endpoints will not allow you to create classic design editor forms and Update a form Application API endpoints will not allow changing a form from responsive to classic. Review the Product Notice for further information and schedule changes.

Bulk API

- Resolved an issue where it was possible to create export definitions with a `dataRetentionDuration` of `null`, resulting in data being deleted sooner than intended. Now, `dataRetentionDuration` can no longer be set to `null`, and the default duration will continue to be 12 hours (ie. `PT12H`).

Platform notices

- Resolved an issue where it was possible to create export definitions with a `dataRetentionDuration` of `null`, resulting in data being deleted sooner than intended. Now, `dataRetentionDuration` can no longer be set to `null`, and the default duration will continue to be 12 hours (ie. `PT12H`).

Release 19D

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 19D.

New features

Application API

- We've added new ways you can filter landing pages when using the Retrieve a list of landing pages endpoint:
    - **Filter by content source**: You can now search for landing pages by content source using the new query parameter `contentSource`, specifying content source by ID:
        - editor = `0`
        - upload = `1`
        - migration = `2`
        - responsive = `3`

      We recommend always using the Application 2.0 API endpoints, as some functionality may not be available in 1.0. The Application 1.0 endpoint cannot retrieve responsive landing pages.

      Example

      Retrieve a list of landing pages, filtered by content source editor (`0`):

      ```
      GET /api/REST/2.0/assets/landingPages?search=contentSource=0&count=1
      ```

      Response body:

```
{
                                                    "elements": [
                                                    {
                                                    "type": "LandingPage",
                                                    "currentStatus": "Draft",
                                                    "id": "8",
                                                    "createdAt": "1419366754",
                                                    "createdBy": "5",
                                                    "depth": "minimal",
                                                    "folderId": "3655",
                                                    "name": "test",
                                                    "permissions": [
                                                    "Retrieve",
                                                    "SetSecurity",
                                                    "Delete",
                                                    "Update"
                                                    ],
                                                    "updatedAt": "1419366754",
                                                    "updatedBy": "5",
                                                    "excludeFromAuthentication": "false",
                                                    "htmlContent": {
                                                    "type": "StructuredHtmlContent",
                                                    "contentSource": "editor"
                                                    }
                                                    }
                                                    ],
                                                    "page": 1,
                                                    "pageSize": 1,
```

```
                "total": 33
}
```

- **Filter by vanity URL**: You can now search for landing pages by the landing page's vanity URL, also known as the `relativePath`. After 19D, using the `search` URL parameter with `name` will include matches for landing page `name` and `relativePath`.

> **Note**: To search on both the `name` and `relativePath`, the leading wildcard "*" is required as the `relativePath` always starts with a "/".

Example

```
GET /api/REST/2.0/assets/landingPages?search=name=*My_Landing_Page
```

OR

```
GET /api/REST/2.0/assets/landingPages?search=*My_Landing_Page
```

Response body:

```
{
                "elements": [
                {
                "type": "LandingPage",
                "currentStatus": "Draft",
```

```
"id": "4699",
"createdAt": "1568155254",
"createdBy": "71",
"depth": "minimal",
"folderId": "1265",
"name": "search_name_and_relativePath"
"permissions": [
"Retrieve",
"SetSecurity",
"Delete",
"Update"
],
"updatedAt": "1568155289",
"updatedBy": "71",
"excludeFromAuthentication": "false",
"htmlContent": {
"type": "RawHtmlContent",
"contentSource": "upload"
},
"relativePath": "/relative_path"
},
{
"type": "LandingPage",
"currentStatus": "Draft",
"id": "4700",
"createdAt": "1568155393",
"createdBy": "71",
"depth": "minimal",
```

```
                "folderId": "1265",
                "name": "Landing_Page_Name",
                "permissions": [
                "Retrieve",
                "SetSecurity",
                "Delete",
                "Update"
                ],
                "updatedAt": "1568155393",
                "updatedBy": "71",
                "excludeFromAuthentication": "false",
                "htmlContent": {
                "type": "RawHtmlContent",
                "contentSource": "upload"
                },
                "relativePath": "/search_name_and_relati
                }
                ],
                "page": 1,
                "pageSize": 1000,
                "total": 2
        }
```


- **Filter by user who created or last updated the landing page**: Use the `search` URL parameter with the `updatedByUserId` or `createdByUserId` properties to filter landing pages.

  Example

  Retrieve a list of landing pages, filtered by the user ID who created the landing page:

```
GET
/api/REST/2.0/assets/landingPages?search=createdByUserId=2&
count=2
```

Response body:


```
{
                                                    "elements": [
                                                    {
                                                    "type": "LandingPage",
                                                    "currentStatus": "Draft",
                                                    "id": "29",
                                                    "createdAt": "1452624350",
                                                    "createdBy": "2",
                                                    "depth": "minimal",
                                                    "folderId": "3655",
                                                    "name": "ttt1 lp",
                                                    "permissions": [
                                                    "Retrieve",
                                                    "SetSecurity",
                                                    "Delete",
                                                    "Update"
                                                    ],
                                                    "updatedAt": "1452624350",
                                                    "updatedBy": "2",
                                                    "excludeFromAuthentication": "false",
                                                    "htmlContent": {
                                                    "type": "StructuredHtmlContent",
```

```
"contentSource": "editor"
}
},
{
"type": "LandingPage",
"currentStatus": "Draft",
"id": "83",
"createdAt": "1530134137",
"createdBy": "2",
"depth": "minimal",
"folderId": "4070",
"name": "Webinar LP",
"permissions": [
"Retrieve",
"SetSecurity",
"Delete",
"Update"
],
"updatedAt": "1530134964",
"updatedBy": "2",
"excludeFromAuthentication": "false",
"htmlContent": {
"type": "ResponsiveHtmlContent",
"contentSource": "responsive"
}
}
],
"page": 1,
```

```
                "pageSize": 2,
                "total": 6
            }
```

- The Retrieve a list of landing pages API endpoint will now return a landing page's vanity URL (`relativePath`) at `minimal` `depth`. For more information, see the product notice.
- You can now search for forms by content source using the new query parameter `contentSource`, specifying content source by ID:
    - classic forms = `0`
    - responsive forms = `3`

    Note that you must specify the content source ID to search.

    Example

    Retrieve a list of classic forms (`0`):

    ```
    GET /api/REST/2.0/assets/forms?search=contentSource=0&count=1
    ```

    Response body:

    ```
    {
                    "elements": [
                    {
                    "type": "Form",
                    "currentStatus": "Draft",
                    "id": "2",
                    "createdAt": "1418404077",
                    "createdBy": "5",
    ```

```
                "depth": "minimal",
                "folderId": "4199",
                "name": "Form 01",
                "permissions": [
                "Retrieve",
                "SetSecurity",
                "Delete",
                "Update"
                ],
                "updatedAt": "1534172012",
                "updatedBy": "9",
                "htmlName": "Fall form 2019",
                "isResponsive": "false"
                }
                ],
                "page": 1,
                "pageSize": 1,
                "total": 43
        }
```


Notice `isResponsive` is set to `false`, indicating the form is a classic form.

Recent changes

Application API

- We've modified the Retrieve a list of activities endpoint so that an email `DeploymentId` now appears in the response for `emailOpen`, `emailSend`, and `emailClickThrough` activity types. The `DeploymentId` allows email activities to be associated with a specific deployment. For more information, see the product notice.

- We've made changes to the Landing Pages endpoints so that `minimal` `depth` will return `relativePath`, and using the `search` URL parameter with `name` when retrieving a list of landing pages will return matches for `name` and `relativePath`. For more information, see the product notice.
- The Form endpoints will now return `htmlName` at `minimal` `depth`. For more information, see the product notice.

Bulk API

- Resolved an issue where creating activity export definitions and setting `maxRecords` would correctly create the definition and respect the `maxRecords` property, however the `maxRecords` property was not appearing when creating or retrieving the definition. The `maxRecords` property now appears as expected in these scenarios.
- Resolved an issue where creating a Bulk API import with the contact fields `{{Contact.Email.IsSubscribed}}` and `{{Contact.Email.IsBounced}}` would result in an incorrect email status. This issue has been resolved.

Platform notices

- The Classic Design Editor changes are coming in our 20A release (February 2020). Review the Product Notice for further information and schedule changes.

Documentation enhancements of note

- The new Bulk API Best Practices section has become a growing list of best practices when using the Bulk API, such as reusing definitions and how to best retrieve data.

Release 19C

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 19C.

New features

Application API

- Added a new endpoint that enables you to retrieve visitor data by visitor GUID, aka `externalId`. This endpoint differs from the existing Retrieve visitor data endpoint, because it is a POST request that can include up to 200 visitors per request. Learn more

  Example

  Retrieve visitor data by visitor GUID:

  ```
  POST /api/REST/2.0/data/visitors
                                          X-HTTP-Method-Override: SEARCH
                                      Content-Type: application/json
  ```

  Request body:

  ```
  {
                                          "externalIds": ["a10746a6-2480-425b-9db7-f20f4ac1c37
  "b50631a2-1234-234b-3ab1-a21f4be1g88z"],
                                          "depth": "minimal"
                                      }
  ```

  Response body:

```
{
                                        "elements": [
                                        {
                                        "type": "VisitorViewData",
                                        "visitorId": "88888",
                                        "createdAt": "1395423247",
                                        "V_IPAddress": "127.0.0.1",
                                        "V_LastVisitDateAndTime": "1395426873",
                                        "externalId": "a10746a6-2480-425b-9db7-f20f4ac1c37e"
                                        "contactId": "8"
                                        },
                                        {
                                        "type": "VisitorViewData",
                                        "visitorId": "12345",
                                        "createdAt": "1395433547",
                                        "V_IPAddress": "128.0.0.1",
                                        "V_LastVisitDateAndTime": "1395426873",
                                        "externalId": "b50631a2-1234-234b-3ab1-a21f4be1g88z",
                                        "contactId": "9"
                                        }
                                        ],
                                        "page": 1,
                                        "pageSize": 5,
                                        "total": 2
                                    }
```

- The Application API Email endpoints have been updated so that `minimal` depth will now return the `subject` property (the email's subject line). Additionally, using the `search` URL parameter with `name` when retrieving a list of emails, will return matches for `name` and `subject`. Learn more

- We've added new query parameters for the Retrieve a list of emails endpoint.

  - `updatedByUserId` - Filters emails by the user ID who last updated the email asset.
  - `createdByUserId` - Filters email by the user ID who created the email asset.

  Example

  Retrieve a list of emails, filtered by the user ID who created the email:

  ```
  GET /api/REST/2.0/assets/emails?search=createdByUserId=2
  ```

  Response body:

  ```
  {
                                  "elements": [
                                  {
                                  "type": "Email",
                                  "currentStatus": "Draft",
                                  "id": "222",
                                  "createdAt": "1525979129",
                                  "createdBy": "2",
                                  "depth": "minimal",
                                  "folderId": "4195",
                                  "name": "Test PMTA 1",
                                  "permissions": [
  ```

```
                "Retrieve",
                "SetSecurity",
                "Delete",
                "Update"
                ],
                "updatedAt": "1525979129",
                "updatedBy": "2",
                "archive": "false",
                "htmlContent": {
                "type": "ResponsiveHtmlContent",
                "contentSource": "responsive"
                },
                "subject": "Test PMTA 1"
                }
                ],
                "page": 1,
                "pageSize": 1000,
                "total": 1
            }
```


See the tutorial for more information about searching.

- You can now search for emails by content source using the new query parameter `contentSource`, specifying content source by ID:
    - editor = `0`
    - upload = `1`
    - responsive = `3`

    Note that you must specify the content source ID to search.

Example

Retrieve a list of emails, filtered by content source editor (`0`):

```
GET /api/REST/2.0/assets/emails?search=contentSource=0&count=1
```

Response body:


```
{
                                        "elements": [
                                        {
                                        "type": "Email",
                                        "currentStatus": "Draft",
                                        "id": "1",
                                        "createdAt": "1418403607",
                                        "depth": "minimal",
                                        "folderId": "3718",
                                        "name": "September Buzz Email",
                                        "permissions": [
                                        "Retrieve",
                                        "SetSecurity",
                                        "Delete",
                                        "Update"
                                        ],
                                        "updatedAt": "1455061318",
                                        "archive": "false",
                                        "htmlContent": {
                                        "type": "StructuredHtmlContent",
                                        "contentSource": "editor"
                                        },
```

```
            "subject": "Test email"
            }
            ...
            "page": 1,
            "pageSize": 1000,
            "total": 56
        }
```


See the tutorial for more information about searching.

Bulk API

- We've added two new activity fields related to campaign responses:
  - `{{Activity.CampaignResponse.CreatedAt}}` - The date and time the campaign response was created.
  - `{{Activity.CampaignResponse.Field(MemberStatus)}}` - The status of the campaign member.

  These new fields enable retrieving campaign response fields with exported activities to analyze campaign response data. These activity fields are available for the Email Open, Email Clickthrough, Email Send, Page View, and Form Submit activity types.

  See an example

  Here's sample export data that includes the new fields:


```
{
            "totalResults": 1,
            "limit": 1000,
            "offset": 0,
            "count": 1,
            "hasMore": false,
```

```
                                    "items": [
                                    {
                                    "ActivityId": "2148269540",
                                    "ActivityType": "EmailSend",
                                    "ActivityDate": "2019-06-11 14:36:46.400",
                                    "ContactId": "1338747",
                                    "EmailRecipientId": "fa00a29f-754a-4ace-9dd9-4129385
                                    "AssetType": "Email",
                                    "AssetName": "Summer Campaign Email",
                                    "AssetId": "9500",
                                    "SubjectLine": "2019 Summer Campaign",
                                    "EmailWebLink":
"http://secure.p03.eloqua.com/e/es.aspx?siteid=426809016&IsAgent=t
rue&e=17800&elq=FA00A29F-754A-4ACE-9DD9-4129385602FB",
                                    "CampaignId": "7198",
                                    "ExternalId": "ESM1P3012148269540",
                                    "DeploymentId": "19493",
                                    "EmailSendType": "Campaign",
                                    "EmailAddress": "api.user@oracle.com",
                                    "ContactIdExt": "CM1P3000001338747",
                                    "CampaignResponseDate": "2019-07-01 14:36:46.400
                                    "CampaignResponseMemberStatus": "Active"
                                    }
                                    ]
                                }
```

Recent changes

Application API

- All API endpoints for emails and landing pages no longer allow creating and editing classic emails and landing pages. When attempting to create or edit a classic email or landing page, the following error will be returned:

  > '410 htmlContent type of StructuredHtmlContent (Classic editor) is no longer supported.'

  For more information about the classic editor sunset, see the Email and Landing Page product notice

- Oracle modified its supported cipher suites used for Transport Layer Security (TLS) connections to Eloqua. This includes programmatic access to Eloqua via APIs. Learn more

Bulk API

- We've modified how Contacts are linked to Bulk API form submit activities, by setting the Contact that's mapped to the Visitor, if there is a Visitor record linked to the form submit activity. For more information, see the product notice.
- Resolved an issue where a Bulk API sync action to globally subscribe a contact was resulting in the contact not appearing in Eloqua's Segment's UI when filtered by date criteria.

Platform notices

- We have revised the dates of the sunsetting of some functionality of our Classic Form Design Editor. While originally we had communicated you would no longer be able to edit or create new forms with the Classic editor as of 19C (August 2019), we have adjusted the date to our 20A release (February 2020). Please review the updated Product Notice for further information and additional schedule changes.

Release 19B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 19B.

New features

Application API

- Asynchronous form processing via API is released under our Controlled Availability program. You can request access to this feature by submitting a request to My Oracle Support. A future product notice will provide details.

- A new property has been added to the Application 1.0 and 2.0 Custom Objects API endpoints. The `deleteLinkedCustomObjectData` property is a boolean that determines whether or not custom object records are deleted when linked contact records are deleted. Does not apply to records that are unmapped or account deletion. Deleting records is irreversible and data cannot be recovered. The default value is `false`. This feature is released under our Controlled Availability program. You can request access to this feature by submitting a request to My Oracle Support.

Bulk API

- We've added two new activity fields: `WebVisitSavedId` and `PageViewSavedId`. `WebVisitSavedId` is available when creating an activity export definition for web visit activity types to include the id of when web visits were saved. `PageViewSavedId` is available when creating an activity export definition for page view activity types to include the id of when page views were saved. You can use the new activity fields in the activity export definition filter to successfully export web visit and page view activities periodically without missing any.

  The syntax for the new activity fields:

- WebVisitSavedId: `{{Activity.Field(WebVisitSavedId)}}`
- PageViewSavedId: `{{Activity.Field(PageViewSavedId)}}`

When export data is retrieved, the `WebVisitSavedId` will be populated with the unique identifier for when the web visit was saved, and `PageViewSavedId` will be populated with the unique identifier for when the page view was saved.

See an example

Here's sample export data that includes `WebVisitSavedId`:


```
{
                                    "ActivityId": "2147573510",
                                    "ActivityType": "WebVisit",
                                    "ActivityDate": "2018-01-19 04:20:58.517",
                                    "ContactId": "1659305",
                                    "VisitorId": "5269",
                                    "VisitorExternalId": "",
                                    "IpAddress": "139.130.4.5",
                                    "WebVisitSavedId": "6110"
                                }
```


Recent changes

Application API

- As part of the Classic Design Editor Sunset, the API endpoints will **not** allow you to create classic emails and landing pages (request body for `htmlContent` with a type of `StructuredHtmlContent`). However, you can still use the API to edit emails and landing pages. Any new emails and landing pages created using the API endpoints should have a request body for `htmlContent` with a type of `RawHtmlContent`. If this is not specified, the

API endpoints will still default to creating new emails and landing pages with `RawHtmlContent`.

Creating emails or landing pages with the `htmlContent` type `ResponsiveHtmlContent` is not supported. If you use the `ResponsiveHtmlContent` type, we cannot guarantee your email or landing page will be responsive.

As of release 19C, all API endpoints for emails and landing pages will not allow creating and editing classic emails and landing pages. Classic forms will still be allowed to be edited through release 20A, but no new classic forms will be allowed to be created starting in 19C. Any forms created or updated with the API endpoints will need to be responsive starting in 19C. A future product notice will provide details on usage.

For more information about the classic editor sunset, see the Email and Landing Page product notice and the Form product notice.

- A new property, `isHidden`, that stores the "Form Visibility" form setting, was added to the Application API Form endpoints. For more information, see the product notice.

Bulk API

- Resolved an issue where performing an activity export for Subscribe, Unsubscribe, and Bounceback activity types using the statement `{{Activity.Field(EmailAddress)}}` could result in email addresses being returned in a case different than the case set on the contact currently. Performing an activity export for these activity types, and using this statement, will now return email addresses in the current case of the contact.

- Resolved an issue where creating an activity import sync and then syncing invalid data to the import definition would result in the sync status being `active` instead of `error`.

  Example

When sync log message "Invalid data.", statusCode "ELQ-00040", occurs with message "Asset Type not found.", statusCode "ELQ-00072", the sync was staying active.


```
{
                                        "items": [
                                        {
                                        "syncUri": "/syncs/399705",
                                        "count": 0,
                                        "severity": "warning",
                                        "statusCode": "ELQ-00123",
                                        "message": "identifierField will be ignored since Activities
updated.",
                                        "createdAt": "2018-12-28T20:44:05.6200000Z"
                                        },
                                        {
                                        "syncUri": "/syncs/399705",
                                        "count": 2,
                                        "severity": "information",
                                        "statusCode": "ELQ-00130",
                                        "message": "Total records staged for import.",
                                        "createdAt": "2018-12-28T20:44:07.1100000Z"
                                        },
                                        {
                                        "syncUri": "/syncs/399705",
                                        "count": 0,
                                        "severity": "information",
                                        "statusCode": "ELQ-00137",
```

```
"message": "Ready for data import processing.",
"createdAt": "2018-12-28T20:44:07.1400000Z"
},
{
"syncUri": "/syncs/399705",
"count": 2,
"severity": "information",
"statusCode": "ELQ-00001",
"message": "Total records processed.",
"createdAt": "2018-12-28T20:44:08.6900000Z"
},
{
"syncUri": "/syncs/399705",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00072",
"message": "Asset Type not found.",
"createdAt": "2018-12-28T20:44:08.9470000Z"
},
{
"syncUri": "/syncs/399705",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00040",
"message": "Invalid data.",
"createdAt": "2018-12-28T20:44:08.9470000Z"
}
],
```

```
        "totalResults": 6,
        "limit": 1000,
        "offset": 0,
        "count": 6,
        "hasMore": false
    }
```

- Optimizations have been made to Bulk API custom object imports that include a large number of fields and records.
- Bulk API Contact export definitions with an EXISTS() Segment filter, e.g. `EXISTS('{{ContactSegment[<id>]}}')`, will only allow up to two system updated or created date criterion. For more information, see the product notice.

Platform notices

- As of release 19C, all API endpoints for emails and landing pages will not allow creating and editing classic emails and landing pages. Classic forms will still be allowed to be edited through release 20A, but no new classic forms will be allowed to be created starting in 19C. Any forms created or updated with the API endpoints will need to be responsive starting in 19C. A future product notice will provide details on usage.

  For more information about the classic editor sunset, see the Email and Landing Page product notice and the Form product notice.

- With the arrival of Eloqua release 19C (Aug 2019), Oracle will be modifying its supported cipher suites used for Transport Layer Security (TLS) connections to Eloqua. This includes programmatic access to Eloqua via APIs. Learn more

Release 19A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 19A.

New features

Application API

- Added a new Audit Log API endpoint to initiate an Eloqua audit log export using the Application API. This API endpoint enables you to specify the type of audit log you want to export within a specified date and time range. If the request is successful, a report will be sent to the email address specified in the request. Previously, initiating audit log exports were only accessible in Eloqua or using an SFTP server after exporting. Learn more

  Example

  Initiate an audit log export for campaign assets for all actions:

  ```
  POST /api/REST/2.0/data/auditLog/reportExport
  Content-Type: application/json
  ```

  Request

  ```
  {
      "auditReportType": "Assets",
      "startDate": "1546837200",
      "endDate": "1547528400",
      "notificationEmail": "API.User@test.com",
      "fileFormat": "DelimitedFile",
      "assetTypes": "Campaigns",
  ```

```
                "actionName": "all"
        }
```

Response

```
201 Created
```

If the request is successful, the exported audit log is sent to the email address specified by `notificationEmail`.

- A new property has been added to the Contact Fields API endpoints. The `showTrustedVisitorsOnly` property is a boolean that determines whether or not a contact field is displayed only to trusted visitors. A trusted visitor is a contact who has undergone an additional form of verification. A contact field where this property is set to `true` will only be displayed to trusted visitors.

  For more information, see the Oracle Eloqua Help Center.

Bulk API

- We've added a new activity field that enables you to filter on the date and time a visitor was linked to a contact. This activity field is available when creating an activity export definition for page view or web visit activity types. Activity records are only returned when linked to a contact. Previously, to export all newly linked activity records, an export as far back as an activity is meaningful would need to be performed to capture newly linked records. With `LinkedToContactDate`, newly linked activity records can efficiently be exported without exporting activities that have already been exported.

  The syntax for the new `LinkedToContactDate` activity field is: `{{Activity.Field(LinkedToContactDate)}}`

  When export data is retrieved, the `LinkedToContactDate` will be populated with the date and time the contact was linked to a visitor.

See an example

Here's sample export data that includes `LinkedToContactDate`:


```
{
                                        "ActivityId": "2147573510",
                                        "ActivityType": "PageView",
                                        "ActivityDate": "2018-01-19 04:20:58.517",
                                        "ContactId": "1659305",
                                        "VisitorId": "5269",
                                        "VisitorExternalId": "",
                                        "WebVisitId": "6448",
                                        "Url":
"http://microsites.eloqua.com/LP=1?PURLCode=&amp;optin=all&amp;D
isplayBanner=1&amp;elqTrackId=xxxxxxxxxxxxx&amp;elqaid=8604&am
p;elqat=1&amp;elqCampaignId=&amp;elq=xxxxxxxx",
                                        "ExternalId": "ABC1P3002147573510",
                                        "LinkedToContactDate": "2017-10-04 05:56:33.177",
                                        "EmailAddress": "API.User@oracle.com",
                                        "ContactIdExt": "CM1P3000001659305"
                                    }
```


Recent changes

- Oracle Eloqua deprecated TLS 1.1 on January 18, 2019. Learn more

Application API

- Resolved an issue with the Application 2.0 Custom Object API endpoints where retrieving and updating a custom object with a custom object field name greater than 50 characters would truncate the `name` value to 50 characters in the response. With release 19A, if the

custom object field name is greater than 50 characters, there will be a validation error.

- The API endpoint to retrieve a list of activities will now only return `EmailContent` when `depth` is set to `complete`. The default `depth` for this endpoint is `complete`, and remains unchanged. Set the `depth` to `minimal` in the request URL to have `EmailContent` not returned. See Request depth for details on setting `depth`.
- Resolved an issue where using the Application 1.0 Accounts API endpoint to find matching accounts based on a search parameter would not return all of the account fields in the response.
- The ability to mark forms as internal so that form submissions are dropped, is now available in Controlled Availability. This feature includes the addition of a new property to the Form API endpoints. You can request access to this feature by submitting a request to My Oracle Support.

Bulk API

- Resolved an issue where creating an activity export definition with `ReferrerUrl` in the filter would cause the export sync to fail. With release 19A, a validation error will be returned if trying to use `ReferrerUrl` in the filter of a page view export definition.
- The Campaign Response endpoints are now Generally Available. These endpoints enable developers to access campaign responses, to support the specific use case of updating a campaign response status in CRM via an App on Program or Campaign Canvas. These endpoints can only be used by an App service instance selected in Response Rules setup, and cannot be accessed otherwise. With this feature, the Campaign Responses Data Source for listeners will also be Generally Available.

  You can learn more about Campaign Responses by reading the tutorial or referring to the API documentation.

- Resolved an issue where a Bulk API import that did not include `C_EmailAddress` within `fields` and `isUpdatingMultipleMatchedRecords` was set to `true`, did not successfully update matching records.

Platform notices

- The API endpoints for emails, landing pages, and forms are undergoing changes in releases 19B and 19C as part of the Classic Design Editor Sunset.

  As of release 19B, the API endpoints will **not** allow you to create classic emails and landing pages (request body for `htmlContent` with a type of `StructuredHtmlContent`). However, you can still use the API to edit emails and landing pages. Any new emails and landing pages created using the API endpoints should have a request body for `htmlContent` with a type of `RawHtmlContent`. Creating emails or landing pages with the `htmlContent` type `ResponsiveHtmlContent` is not supported. If you use the `ResponsiveHtmlContent` type, we cannot guarantee your email or landing page will be responsive. For the forms API endpoints, you will still be able to use the API to create and update classic forms.

  As of release 19C, all API endpoints will **not** allow creating and editing classic emails, landing pages, or forms. Any forms created with the API endpoints will need to be responsive. A future product notice will provide details on usage.

  For more information about the classic editor sunset, see the product notice.

Documentation enhancements of note

- We've added a new tutorial that explains how to use the Campaign API endpoints to create a detailed campaign with steps. Learn more
- There's a new campaign element reference page. This topic lists the JSON objects needed to create campaign steps when creating campaigns using the Campaigns API endpoints. Learn more

Release 18D

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 18D.

New features

Application API

- You can now use the Landing Pages Application API 2.0 endpoints to create, update, retrieve, and delete landing pages. Learn more
- You can now use the Forms Application API 2.0 endpoints to create, update, retrieve, and delete forms. Learn more
- You can now use the Form Data Application API 2.0 endpoints to create, retrieve, and delete form data. Learn more

Recent changes

Authentication

- Resolved an issue where calls to Eloqua's OAuth 2.0 token endpoint would return an empty response body if there was an error with the request. Errored requests made to Eloqua's OAuth 2.0 token endpoint will now return a detailed message. The token endpoint is `https://login.eloqua.com/auth/oauth2/token`.
- We modified the Eloqua OAuth 2.0 authorization flow initiation to only accept one initiation per minute for any given user of an app. For more information, see the product notice on Topliners.

Application API

- The `processingStepErrors` property has been added to the response to submitting a form using the Application 1.0 and 2.0 endpoints. This property was previously in Early Preview status in Release 18C. This property provides more detailed information about form processing step errors. For more information, see the product notice on Topliners.

  Example

  Create new form data for the form asset with id #4040:

  ```
  POST /API/REST/2.0/data/form/4040
                                  Content-Type: application/json
  ```

  Request body:

  ```
  {
                                      "fieldValues": [
                                      {
                                      "type": "FieldValue",
                                      "id": "25093",
                                      "value": "John"
                                      }
                                      ]
                                  }
  ```

  Request response:

  ```
  {
                                      "id": "94486",
                                      "fieldValues": [
  ```

```
                                {
                                "type": "FieldValue",
                                "id": "25093",
                                "value": "John"
                                }
                                ],
                                "processingStepErrors": [
                                {
                                "type": "ProcessingStepError",
                                "errorCode": 1001,
                                "errorDescription": "EmailAddress must be a valid email a
                                "errorMessage": "ProcessingStep: Validation Error",
                                "processingStepDisplayName": "Create / Update Contact
or Company",
                                "processingStepId": "10119",
                                "processingStepTypeId": "2000"
                                }
                                ]
                        }
```


The error indicates `EmailAddress` was not included in the request, but is required to complete one or more processing steps.

- A new form field validation property has been added to the Application 2.0 Form endpoints called `PreventXSSCondition`. This field validation is being added to prevent form data from being saved if HTML is present in a field. For more information, see the product notice on Topliners.

- The `requirement` property has been added to the response when submitting form data with invalid values using the Application 1.0 and 2.0 endpoints. This property provides more information about invalid form data being submitted to a form field. See the Eloqua Help Center for more information about form field validation.

  Example: Submitting form data

  Submit form data when a field with IsDateCondition validation contains an invalid value:

```
POST /API/REST/2.0/data/form/4
                                    Content-Type: application/json
```

  Request body:

```
{
                                        "type": "FormData",
                                        "fieldValues": [
                                        {
                                        "type": "FieldValue",
                                        "id": "2",
                                        "name": "emailAddress",
                                        "value": "jon.doe@gmail.com"
                                        },
                                        {
                                        "type": "FieldValue",
                                        "id": "4",
                                        "name": "leadScoreDateMostRecent1",
                                        "value": "xyz"
```

```
                    }
                    ]
                }
```

Response prior to 18D:

```
[
                    {
                    "type": "ObjectValidationError",
                    "container": {
                    "type": "ObjectKey",
                    "objectType": "FormData"
                    },
                    "property": "1",
                    "value": " xyz "
                    }
                ]
```

Response after 18D:

```
[
                    {
                    "type": "ObjectValidationError",
                    "container": {
                    "type": "ObjectKey",
                    "objectType": "FormData"
                    },
```

```
                "property": "4",
                "requirement": {
                "type": "DateRequirement"
                },
                "value": "xyz"
                }
                ]
```

The response indicates the value "xyz" does not meet the DateRequirement condition.

- Resolved an issue where a campaign import could modify the end date of a campaign that is Completed or Active, causing campaigns to end before the intended end date. Campaign imports will no longer be able to modify the end date of campaigns that are Completed or Active.
- As mentioned in the Eloqua 18C changelog, the `resendLimit` property introduced in 18A was removed from the `processingSteps` type `FormStepSendEmail` for Application API Form endpoints. For more information, see the product notice on Topliners.
- As mentioned in the Eloqua 18C changelog, the Create an external activity Application API endpoint will no longer create External Asset Types or External Activity Types if they do not exist. A new Action Permission, “Register External Activities”, is required to use the Create an external activity Application API endpoint. Learn more

Bulk API

- You can now include `Contact.Id` for Bounceback, Subscribe, and Unsubscribe activity export definitions. This enhancement enables developers to include `Contact.Id` on export

definitions for all activity types.

Example: Create an activity export

Create a bounceback activity export definition:

```
POST /api/bulk/2.0/activities/exports
                                    Content-Type: application/json
```

Request body:

```
{
                                        "filter": "'{{Activity.Type}}'='Bounceback'",
                                        "name": "Bulk Activity Export - Bounceback",
                                        "fields": {
                                        "ActivityId": "{{Activity.Id}}",
                                        "ActivityType": "{{Activity.Type}}",
                                        "ActivityDate": "{{Activity.CreatedAt}}",
                                        "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                                        "ContactId": "{{Activity.Contact.Id}}",
                                        "AssetType": "{{Activity.Asset.Type}}",
                                        "AssetName": "{{Activity.Asset.Name}}",
                                        "AssetId": "{{Activity.Asset.Id}}",
                                        "CampaignId": "{{Activity.Campaign.Id}}",
                                        "CRMCampaignId": "{{Activity.Campaign.Field(CRMCamp
                                        "CampaignName": "{{Activity.Campaign.Field(Campaign
                                        "ExternalId": "{{Activity.ExternalId}}",
                                        "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                                        "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
```

```
                "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
                "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
                "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
                }
            }
```

Request response:

```
{
                "name": "Bulk Activity Export - Bounceback",
                "fields": {
                "ActivityId": "{{Activity.Id}}",
                "ActivityType": "{{Activity.Type}}",
                "ActivityDate": "{{Activity.CreatedAt}}",
                "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                "ContactId": "{{Activity.Contact.Id}}",
                "AssetType": "{{Activity.Asset.Type}}",
                "AssetName": "{{Activity.Asset.Name}}",
                "AssetId": "{{Activity.Asset.Id}}",
                "CampaignId": "{{Activity.Campaign.Id}}",
                "CRMCampaignId": "{{Activity.Campaign.Field(CRMCamp
                "CampaignName": "{{Activity.Campaign.Field(Campaign
                "ExternalId": "{{Activity.ExternalId}}",
                "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
                "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
                "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
```

```
        },
        "filter": "'{{Activity.Type}}'='Bounceback'",
        "dataRetentionDuration": "PT12H",
        "uri": "/activities/exports/76",
        "createdBy": "API.User",
        "createdAt": "2018-10-10T13:49:44.7566016Z",
        "updatedBy": "API.User",
        "updatedAt": "2018-10-10T13:49:44.7566016Z"
    }
```


- You can now include user fields for EmailOpen, EmailClickthrough, and EmailSend activity export definitions. One use case this enhancement enables, is allowing including sender and user attributes on email activities exported via the Bulk API, so that the activity can be properly assigned to the correct user in CRM.

  Discover user fields using the retrieve a list of user fields endpoint (`GET /api/bulk/2.0/users/fields`). This endpoint will be documented in a future release.

  User fields can be added with the following statement: `{{Activity.User.Field(<Field Name>)}}`

  See a list of user fields, by Field Name, that can be included

  - UserId (Eloqua User ID)
  - Signature Layout Fields
    - Company (Company Display Name)
    - CompanyURL (Company Website URL)
    - JobTitle (Job Title)
    - Phone (Business Phone)
    - CellPhone (Mobile Phone)

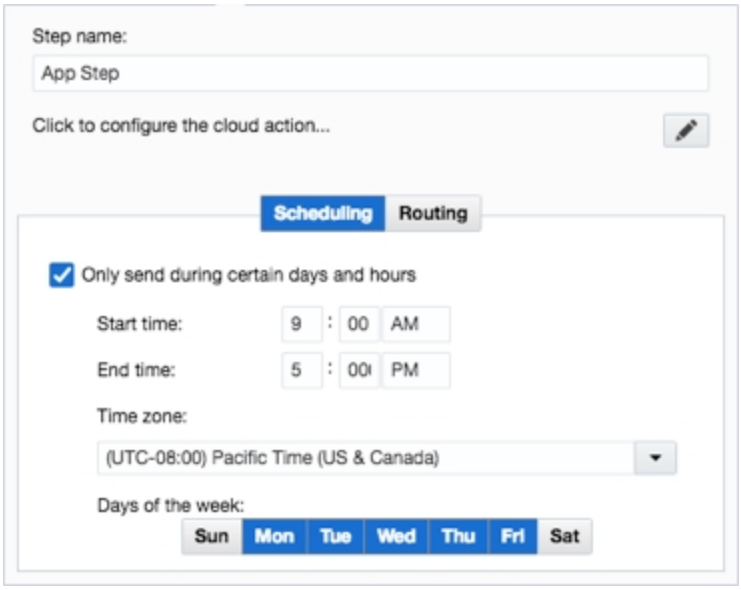
        - SenderDisplayName (Email Sender Display Name)
        - ReplyToAddress (Email Reply-To Address)
        - Department (Department)
        - Fax (Fax Number)
        - Address1 (First Address)
        - Address2 (Second Address)
        - City (City)
        - State (State)
        - Country (Country)
        - Zip (Zip/Postal Code)
        - EmailSenderAddress (Email Sender Address)
        - PersonalURL (Personal Website URL)
        - PersonalMessage (Personal Message)
    - CRM User IDs
        - CRMSfdcUserId (Salesforce User ID)
        - CRMMSDUserId (MS Dynamics User ID)
        - CRMOSCUserId (Oracle Sales Cloud User ID)
        - CRMSODUserId (Siebel User ID)

Platform notices

App Developer Framework

- There is now a **Scheduling** option available for app action steps on Campaign and Program Canvases. If a user sets certain days and hours under **Scheduling**, records that enter the step outside of the selected days and hours, will remain in an “Awaiting action” status until within the selected days and hours, at which time the Notification URL call will then be sent to the app.

See an example of how the Scheduling option will look



- The ability to mark forms as internal so that form submissions are dropped, will be in Controlled Availability in Eloqua Release 18D and 19A. This feature includes the addition of a new property to Form API endpoints. You can request access to this feature by submitting a request to My Oracle Support.
- Oracle Eloqua will be deprecating TLS 1.1 on January 18, 2019. Learn more

Documentation enhancements of note

- We've added a new topic about Bulk API data types and how to retrieve more information about Bulk API data types. Learn more
- We've added a new tutorial that explains how to use cURL to upload data to the Bulk API from a file on your local machine such as a .json or .csv file. Learn more

Release 18C

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 18C.

New features

App Developer Framework

- We've added a new endpoint to enable developers to retrieve feeder service instances. Retrieve up to 200 feeder service instances, searched by service instance GUID. For more information, see Retrieves information for up to 200 feeder service instances, searched by service instance GUID..

  Example

  Retrieve 2 feeder service instances:

  ```
  POST api/cloud/1.0/feeders/instances
  X-HTTP-Method-Override: SEARCH
  Content-Type: application/json
  ```

  Request body:

  ```
  {
  "ids": [
  "4de74ef1-bb2c-44d6-97ba-6acb1565bc58",
  "97e3f90e-6ef8-40a8-9554-588a72de440e"
  ]
  }
  ```

Important: Remember to include the dashes for the service instance GUIDs. A 400 validation error will be returned if the dashes are omitted.

Request response:


```
{
                                        "items": [
                                        {
                                        "statement": "{{FeederInstance
(4de74ef1bb2c44d697ba6acb1565bc58)}}",
                                        "dependentAssetType": "Campaign",
                                        "dependentAssetId": 6662,
                                        "requiresConfiguration": true,
                                        "uri":
"/feeders/90289/instances/4de74ef1bb2c44d697ba6acb1565bc58",
                                        "createdBy": "API.User",
                                        "createdAt": "2018-06-01T17:57:00.1470000Z",
                                        "updatedBy": "API.User",
                                        "updatedAt": "2018-06-01T17:57:23.5130000Z"
                                        },
                                        {
                                        "statement": "{{FeederInstance
(97e3f90e6ef840a89554588a72de440e)}}",
                                        "dependentAssetType": "Program",
                                        "dependentAssetId": 1385,
                                        "requiresConfiguration": true,
                                        "uri":
```

```
"/feeders/90289/instances/97e3f90e6ef840a89554588a72de440e",
                                        "createdBy": "API.User",
                                        "createdAt": "2018-06-01T18:10:21.5700000Z",
                                        "updatedBy": "API.User",
                                        "updatedAt": "2018-06-01T18:10:36.8770000Z"
                                        }
                                        ]
                                    }
```

Application API

- A new property `processingStepErrors` has been added to the response to submitting a form (POST /api/REST/1.0/data/form/<formId>). This feature is available for Early Preview in 18C, and will be generally available with 18D. You can request access to this feature by submitting a request to My Oracle Support.

  Example

  Create some new form data for the form asset with id #4040:

  ```
  POST /api/rest/1.0/data/form/4040
                                      Content-Type: application/json
  ```

  Request body:

  ```
  {
                                        "fieldValues": [
                                        {
                                        "type": "FieldValue",
                                        "id": "25093",
  ```

```
                                        "value": "John"
                                        }
                                        ]
                                }
```

Request response:

```
{
                                        "id": "94486",
                                        "fieldValues": [
                                        {
                                        "type": "FieldValue",
                                        "id": "25093",
                                        "value": "John"
                                        }
                                        ],
                                        "processingStepErrors": [
                                        {
                                        "type": "ProcessingStepError",
                                        "errorCode": 1001,
                                        "errorDescription": "EmailAddress must be a valid email a
                                        "errorMessage": "ProcessingStep: Validation Error",
                                        "processingStepDisplayName": "Create / Update Contac
or Company",
                                        "processingStepId": "10119",
                                        "processingStepTypeId": "2000"
                                        }
```

```
                ]
            }
```

The error indicates `EmailAddress` was not included in the request, but is required to complete one or more processing steps.

- You can now retrieve a list of campaigns using the `search` parameter with the `startAt` parameter to filter campaigns by campaign activation date.

  Example

  Retrieve a list of campaigns:

  ```
  GET /api/rest/2.0/assets/campaigns?search=startAt<1530466857&depth=partial&count=1
  ```

  Request response:

  ```
  {
                  "elements": [
                  {
                  "type": "Campaign",
                  "currentStatus": "Completed",
                  "id": "76",
                  "createdAt": "1488209626",
                  "createdBy": "17",
                  "depth": "partial",
                  "folderId": "3817",
  ```

```
"name": "A/B testing campaign 02272017",
"permissions": [
"Retrieve",
"SetSecurity",
"Delete",
"Update",
"Activate"
],
"updatedAt": "1488209649",
"updatedBy": "17",
"actualCost": "0.00",
"budgetedCost": "0.00",
"campaignCategory": "emailMarketing",
"campaignType": "",
"crmId": "",
"endAt": "1495899249",
"fieldValues": [
{
"type": "FieldValue",
"id": "9",
"value": ""
},
{
"type": "FieldValue",
"id": "10",
"value": ""
},
{
```

```
                "type": "FieldValue",
                "id": "11",
                "value": ""
                }
                ],
                "firstActivation": "1488209649",
                "isEmailMarketingCampaign": "true",
                "isIncludedInROI": "true",
                "isSyncedWithCRM": "true",
                "product": "",
                "region": "",
                "startAt": "1488209649"
                }
                ],
                "page": 1,
                "pageSize": 1,
                "total": 45
            }
```


Bulk API

- We've added new endpoints to enable developers to retrieve activity fields. Refer to the API documentation to learn how you can retrieve all activity fields, or retrieve an activity field by id.

  Example: Retrieve all activity fields

  Retrieve the first two activity fields in your database:

  ```
  GET /api/bulk/2.0/activities/fields?limit=2
  ```

Request response:


```
{
                                        "items": [
                                        {
                                        "name": "Activity Type",
                                        "internalName": "ActivityType",
                                        "dataType": "string",
                                        "hasReadOnlyConstraint": true,
                                        "hasNotNullConstraint": true,
                                        "hasUniquenessConstraint": false,
                                        "statement": "{{Activity.Type}}",
                                        "activityTypes": [
                                        "EmailSend",
                                        "EmailOpen",
                                        "EmailClickthrough",
                                        "Subscribe",
                                        "Unsubscribe",
                                        "Bounceback",
                                        "FormSubmit",
                                        "PageView",
                                        "WebVisit"
                                        ],
                                        "uri": ""
                                        },
                                        {
                                        "name": "Id",
                                        "internalName": "Id",
```

```
            "dataType": "number",
            "hasReadOnlyConstraint": true,
            "hasNotNullConstraint": true,
            "hasUniquenessConstraint": false,
            "statement": "{{Activity.Id}}",
            "activityTypes": [
            "WebVisit",
            "PageView",
            "FormSubmit",
            "Bounceback",
            "Unsubscribe",
            "Subscribe",
            "EmailClickthrough",
            "EmailOpen",
            "EmailSend"
            ],
            "uri": ""
            }
            ],
            "totalResults": 32,
            "limit": 2,
            "offset": 0,
            "count": 2,
            "hasMore": true
        }
```

- You can now include campaign fields in activity export definitions. This enhancement enables developers to properly assign an activity to the correct campaign when data is exported from Eloqua to an external system.

  Example: Create an activity export

  Create a new activity export definition:

```
POST /api/bulk/2.0/activities/exports
                                    Content-Type: application/json
```

  Request body:

```
{
                                        "filter": "'{{Activity.Type}}'='EmailOpen'",
                                        "name": "Bulk Activity Export - Email Open",
                                        "fields": {
                                        "ActivityId": "{{Activity.Id}}",
                                        "ActivityType": "{{Activity.Type}}",
                                        "ActivityDate": "{{Activity.CreatedAt}}",
                                        "ContactId": "{{Activity.Contact.Id}}",
                                        "IpAddress": "{{Activity.Field(IpAddress)}}",
                                        "VisitorId": "{{Activity.Visitor.Id}}",
                                        "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
                                        "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                                        "AssetType": "{{Activity.Asset.Type}}",
                                        "AssetName": "{{Activity.Asset.Name}}",
                                        "AssetId": "{{Activity.Asset.Id}}",
                                        "SubjectLine": "{{Activity.Field(SubjectLine)}}",
                                        "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
```

```
                                    "CampaignId": "{{Activity.Campaign.Id}}",
                                    "CRMCampaignId": "{{Activity.Campaign.Field
(CRMCampaignId)}}",
                                    "CampaignName": "{{Activity.Campaign.Field
(CampaignName)}}",
                                    "ExternalId": "{{Activity.ExternalId}}",
                                    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                                    "EmailSendType": "{{Activity.Field(EmailSendType)}}",
                                    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                                    "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
                                    }
                                }
```

Request response:

```
{
                                    "name": "Bulk Activity Export - Email Open",
                                    "fields": {
                                    "ActivityId": "{{Activity.Id}}",
                                    "ActivityType": "{{Activity.Type}}",
                                    "ActivityDate": "{{Activity.CreatedAt}}",
                                    "ContactId": "{{Activity.Contact.Id}}",
                                    "IpAddress": "{{Activity.Field(IpAddress)}}",
                                    "VisitorId": "{{Activity.Visitor.Id}}",
                                    "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
                                    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                                    "AssetType": "{{Activity.Asset.Type}}",
                                    "AssetName": "{{Activity.Asset.Name}}",
```

```
                "AssetId": "{{Activity.Asset.Id}}",
                "SubjectLine": "{{Activity.Field(SubjectLine)}}",
                "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
                "CampaignId": "{{Activity.Campaign.Id}}",
                "CRMCampaignId": "{{Activity.Campaign.Field
(CRMCampaignId)}}",
                "CampaignName": "{{Activity.Campaign.Field
(CampaignName)}}",
                "ExternalId": "{{Activity.ExternalId}}",
                "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                "EmailSendType": "{{Activity.Field(EmailSendType)}}",
                "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
                },
                "filter": "'{{Activity.Type}}'='EmailOpen'",
                "dataRetentionDuration": "PT12H",
                "uri": "/activities/exports/66",
                "createdBy": "API.User",
                "createdAt": "2018-07-17T14:50:52.3183789Z",
                "updatedBy": "API.User",
                "updatedAt": "2018-07-17T14:50:52.3183789Z"
            }
```


Recent changes

- The Campaign Response endpoints are now generally optional. These endpoints enable developers to access campaign responses, to support the specific use case of updating a campaign response status in CRM via an App on Program or Campaign Canvas. These

endpoints can only be used by an App service instance selected in Response Rules setup, and cannot be accessed otherwise. Submit a request to My Oracle Support to start using this feature. You can learn more about Campaign Responses by reading the tutorial or referring to the API documentation.

- Optimizations have been made to `EmailOpen` and `EmailClickthrough` Bulk API exports that include Activity Date in the filter.
- The “Manage Contact Fields” Action Permission is now required to create, update, and delete Contacts via the Contact fields Application API endpoints. Learn more
- The "Define External Assets and Activities" Action Permission is now required to create, update, or delete External Assets and External Asset Types using the Application API endpoints. Learn more
- Access to Eloqua's Web Services Description Language (WSDL) is now available only by submitting a request to My Oracle Support.

Documentation enhancements of note

- The tutorial on how to use the search URL parameter now includes information on best practices when using a wildcard character. Learn more

Platform notices

- With the arrival of Eloqua release 18D (Nov 16 - 17, 2018), the `resendLimit` property introduced in 18A will be removed from the `processingSteps` type `FormStepSendEmail` for Application API Form endpoints.
- With the arrival of Eloqua release 18D (Nov 16 - 17, 2018), the Create an external activity Application API endpoint will no longer create External Asset Types or External Activity Types if they do not exist. Learn more

Release 18B

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 18B.

New features

Application API

- You can now use the Programs API in REST 2.0 to create, update, retrieve, delete, activate, deactivate, and pause Programs. Learn more.

Bulk API

- We've added opportunities endpoints that enable developers to import opportunities into Eloqua's opportunity object. Developers can then perform another import to link the opportunities with Eloqua contacts. For more information on these endpoints, see the API documentation or the tutorial.

  Example: Create an opportunity import definition

  Create an opportunity import definition:

```
POST /opportunities/imports
                                        Content-Type: application/json
```

  Request body:

```
{
                                            "name": "Opportunity Import",
                                            "fields": {
                                            "OpportunityID": "{{Opportunity.Id}}",
```

```
                "OpportunityName": "{{Opportunity.Field(Name)}}",
                "AccountName": "{{Opportunity.Field(AccountName)}}",
                "CreatedDate": "{{Opportunity.CreatedAt}}",
                "Amount": "{{Opportunity.Field(Amount)}}",
                "CloseDate": "{{Opportunity.Field(CloseDate)}}",
                "Currency": "{{Opportunity.Field(Currency)}}",
                "ForecastToCloseDate": "{{Opportunity.Field
(ForecastToCloseDate)}}",
                "Stage": "{{Opportunity.Field(Stage)}}",
                "Territory": "{{Opportunity.Field(Territory)}}"
                },
                "identifierFieldName": "OpportunityID",
                "isIdentifierFieldCaseSensitive": false
            }
```

Request response:

```
{
                "isIdentifierFieldCaseSensitive": false,
                "name": "Opportunity Import",
                "fields": {
                "OpportunityID": "{{Opportunity.Id}}",
                "OpportunityName": "{{Opportunity.Field(Name)}}",
                "AccountName": "{{Opportunity.Field(AccountName)}}",
                "CreatedDate": "{{Opportunity.CreatedAt}}",
                "Amount": "{{Opportunity.Field(Amount)}}",
                "CloseDate": "{{Opportunity.Field(CloseDate)}}",
                "Currency": "{{Opportunity.Field(Currency)}}",
```

```
                                    "ForecastToCloseDate": "{{Opportunity.Field
(ForecastToCloseDate)}}",
                                    "Stage": "{{Opportunity.Field(Stage)}}",
                                    "Territory": "{{Opportunity.Field(Territory)}}"
                                    },
                                    "identifierFieldName": "OpportunityID",
                                    "isSyncTriggeredOnImport": false,
                                    "dataRetentionDuration": "P7D",
                                    "uri": "/opportunities/imports/24",
                                    "createdBy": "API.User",
                                    "createdAt": "2018-02-15T15:28:32.4730364Z",
                                    "updatedBy": "API.User",
                                    "updatedAt": "2018-02-15T15:28:32.4730364Z"
                                }
```

Example: Create a new opportunity contact linkage import definition

Create a new opportunity contact linkage import definition:

```
POST /opportunities/contacts/imports
                                    Content-Type: application/json
```

Request body:

```
{
                                    "name": "Opportunity Contact Linkage Import",
                                    "fields": {
                                    "EmailAddress": "{{Opportunity.Contact.Field(C_EmailAd
                                    "OpportunityID": "{{Opportunity.Id}}"
```

```
                    },
                    "linkOpportunitiesCaseSensitiveMatchField": false,
                    "linkOpportunitiesCaseSensitiveSourceField": false,
                    "linkOpportunitiesEntityType": "Contact",
                    "linkOpportunitiesMatchFieldName": "OpportunityID",
                    "linkOpportunitiesMultipleSourceMatches": true,
                    "linkOpportunitiesSourceField": "EmailAddress"
                }
```

Request response:

```
{
                    "linkOpportunitiesMatchFieldName": "OpportunityID",
                    "linkOpportunitiesSourceField": "EmailAddress",
                    "linkOpportunitiesEntityType": "Contact",
                    "linkOpportunitiesCaseSensitiveSourceField": false,
                    "linkOpportunitiesCaseSensitiveMatchField": false,
                    "linkOpportunitiesMultipleSourceMatches": true,
                    "name": "Opportunity Contact Linkage Import",
                    "fields": {
                    "EmailAddress": "{{Opportunity.Contact.Field(C_EmailAd
                    "OpportunityID": "{{Opportunity.Id}}"
                    },
                    "identifierFieldName": "OpportunityID",
                    "isSyncTriggeredOnImport": false,
                    "dataRetentionDuration": "P7D",
                    "uri": "/opportunities/contacts/imports/23",
                    "createdBy": "API.User",
```

```
                "createdAt": "2018-02-15T15:03:43.3345307Z",
                "updatedBy": "API.User",
                "updatedAt": "2018-02-15T15:03:43.3345307Z"
            }
```

- We've added a new parameter, `nullIdentifierFieldName`, that enables you to null the `identifierFieldName` when performing an account or contact import.

  Example: Create a contact import definition with the nullIdentifierFieldName parameter

  Create a contact import definition:

```
POST api/bulk/2.0/contacts/imports
                Content-Type: application/json
```

  Request body:

```
{
                "name": "Contact Import - Null Identifier Field",
                "fields": {
                "C_SFDCContactID": "{{Contact.Field(C_SFDCContactID)}
                },
                "identifierFieldName": "C_SFDCContactID",
                "nullIdentifierFieldName": true
            }
```

  Request response:

```
{
                "name": "Contact Import - Null Identifier Field",
```

```
        "fields": {
        "C_SFDCContactID": "{{Contact.Field(C_SFDCContactID)}}
        },
        "identifierFieldName": "C_SFDCContactID",
        "isSyncTriggeredOnImport": false,
        "dataRetentionDuration": "P7D",
        "isUpdatingMultipleMatchedRecords": false,
        "nullIdentifierFieldName": true,
        "uri": "/contacts/imports/35",
        "createdBy": "API.User",
        "createdAt": "2018-04-06T13:06:04.9178388Z",
        "updatedBy": "API.User",
        "updatedAt": "2018-04-06T13:06:04.9178388Z"
}
```


Recent changes

- Prior to release 18B, it was possible for email sends to fail if an email was sent to deleted contacts. As of release 18B when emails are sent via any channel in Eloqua and one or more of the target contacts are in a deleted state, emails sent to that recipient will fail and a log entry is saved indicating why the email send failed.
- Resolved an issue where retrieving a list of accounts in REST 1.0 wasn't returning values for custom account fields.
- Resolved an issue where it was possible to update completed campaigns. Campaigns that are completed can no longer be updated.
- As announced in Code It, the `recordCount` property has been removed. Learn more

Documentation enhancements of note

- We've added a new tutorial to walkthrough how to use the `search` URL parameter for Eloqua's Application API. This tutorial contains usage details about the parameter and various examples about how the parameter can be used. Learn more
- Added a new tutorial to walkthrough how to export all assets in an Eloqua database using the Application API. Learn more
- Added `viewId` to the URL parameter list. Use the `viewId` parameter to specify a contact or account view to filter data according to that view. Learn more
- Added an example that explains how to create a campaign with a custom end date. Learn more

Release 18A

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 18A.

New features

Application API

- We've added a new property, `resendLimit`, to the Application API Form endpoints when there is a `FormStepSendEmail` within `processingSteps`. This property controls the amount of times a form will attempt to resend an email. Learn more

Recent changes

Application API

- Resolved an issue with the Retrieve visitor data Application API endpoint where `V_TimeZone` was returned inversed. In example, if the offset was `UTC+5`, it returned `UTC-5`.

- Resolved an issue in REST 1.0 where creating or updating an account that contained custom account fields would result in the account being created or updated, but without the custom account fields.

  Example

  Creating an account with custom account fields prior to release 18A resulted in the custom field values being ignored and not set:

  Request

  ```
  POST /api/rest/1.0/data/account
  ```

  Request body

  ```
  {
                                  "name": "Account with custom account fields",
                                  "address2": "123 Redwood Shores",
                                  "fieldValues": [
                                  {
                                  "type": "FieldValue",
                                  "id": "100187",
                                  "value": "1"
                                  },
                                  {
                                  "type": "FieldValue",
                                  "id": "100193",
                                  "value": "1.5000"
                                  },
                                  {
                                  "type": "FieldValue",
  ```

```
                "id": "100212",
                "value": "123.1230"
                }
                ]
        }
```

Response body

```
{
                "type": "Account",
                "id": "22",
                "createdAt": "1517415972",
                "depth": "complete",
                "description": "",
                "name": "Account with custom account fields",
                "updatedAt": "1517415972",
                "address1": "",
                "address2": "123 Redwood Shores",
                "address3": "",
                "businessPhone": "",
                "city": "",
                "country": "",
                "fieldValues": [],
                "postalCode": "",
                "province": ""
        }
```

Performing the same request in release 18A will now successfully create the account with the custom account fields.

Response body


```
{
                                        "type": "Account",
                                        "id": "22",
                                        "createdAt": "1517415972",
                                        "depth": "complete",
                                        "description": "",
                                        "name": "Account with custom account fields",
                                        "updatedAt": "1517415972",
                                        "address1": "",
                                        "address2": "123 Redwood Shores",
                                        "address3": "",
                                        "businessPhone": "",
                                        "city": "",
                                        "country": "",
                                        "fieldValues": [
                                        {
                                        "type": "FieldValue",
                                        "id": "100187",
                                        "value": "1"
                                        },
                                        {
                                        "type": "FieldValue",
                                        "id": "100193",
                                        "value": "1.5000"
                                        },
                                        {
                                        "type": "FieldValue",
```

```
                "id": "100212",
                "value": "123.1230"
                }
                ],
                "postalCode": "",
                "province": ""
            }
```


Bulk API

- Resolved an issue where Bulk API activity exports including contact fields were failing when Label-Based Access Control was enabled.
- Improved validation for Bulk API activity exports to return a validation error if including contact fields without an `Activity` root. There was validation missing which allowed field statements to be used in activity exports such as:

  `{{Contact.Field(<contact_field_name>)}}`

  This is incorrect, and will now throw a validation error. The correct syntax for using a contact field statement in an activity export is:

  `{{Activity.Contact.Field(<contact_field_name>)}}`

Platform notices

- Emails created in the New Design Editor, Generally Available in Release 18A, will only be accessible via the Email 2.0 endpoints.

Release 493

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 493.

New features

Bulk API

- We've added new endpoints to enable developers to access campaign responses, to support the specific use case of updating a campaign response status in CRM via an App on Program or Campaign Canvas. These endpoints can only be used by an App service instance selected in Response Rules setup, and cannot be accessed otherwise. This feature is currently in Controlled Availability. Learn more

Application API

- You can now use the Emails API in REST 2.0 to create, update, retrieve, and delete emails. Learn more.

Recent changes

Application API

- Resolved an issue in REST 2.0 where retrieving custom object data would return incorrect date values. For example, a custom object field value saved as 2017-03-04 should return epoch time of March 4, 2017, but instead returned epoch time of April 3, 2017 when the output format is set to dd/mm/yyyy.
- Resolved two issues in REST 2.0 where creating a custom object instance would return an incorrect value for `name`, and not display all fields.
- Resolved two issues in REST 2.0 where retrieving a custom object instance did not return the `depth` or `isMapped` properties.
- Resolved an issue in REST 2.0 where retrieving custom object data and filtering by custom object field would result in an error if the field name started with a number.
- Resolved an instance where it was possible for an email deployment to fail if a single email deployment was being sent to multiple contacts with the same email address. This issue only

affected instances where contacts can exist without an email address and duplicate email addresses are enabled.

Bulk API

- Prior to release 493, email activities exported via the Bulk API would include the current subject line of the email. With 493, email activities exported via the Bulk API will include the subject line at the time of email send.
- When viewing dependencies for contact and account fields in the Fields and Views area of Eloqua, Bulk API Import and Export dependencies are now displayed within the dependency checker.

Release 492

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 492.

New features

Bulk API

- The `/api/bulk/2.0/syncs/{id}/logs` endpoint has been improved to more efficiently handle fields included in data upload that are not included in an import definition. Previously, the sync logs would display a separate message per field not included in the import definition, per record it's included in. With this update, a field is only mentioned once in sync log messaging even if the field is included multiple times.

  Example: Retrieving a sync's logs prior to Release 492

  Contact import definition:

```
{
                                    "name": "Example Import",
                                    "fields": {
                                    "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
                                    },
                                    "identifierFieldName": "EmailAddress",
                                    "syncActions": [],
                                    "isSyncTriggeredOnImport": false,
                                    "dataRetentionDuration": "P7D",
                                    "isUpdatingMultipleMatchedRecords": false,
                                    "uri": "/contacts/imports/47",
                                    "createdBy": "API.User",
                                    "createdAt": "2017-02-14T23:43:20.4570000Z",
                                    "updatedBy": "API.User",
                                    "updatedAt": "2017-02-14T23:43:20.4570000Z"
                                }
```


Uploaded data:


```
[
                                    {
                                    "EmailAddress": "fieldnotindef1@test.elq",
                                    "notindeffield1": "1"
                                    },
                                    {
                                    "EmailAddress": "fieldnotindef2@test.elq",
                                    "notindeffield1": "1",
                                    "notindeffield2": "2"
                                    },
```

```
                {
                "EmailAddress": "fieldnotindef3@test.elq",
                "notindeffield1": "1",
                "notindeffield2": "2",
                "notindeffield3": "3"
                },
                {
                "EmailAddress": "fieldnotindef4@test.elq",
                "notindeffield1": "1",
                "notindeffield2": "2",
                "notindeffield3": "3",
                "notindeffield4": "4"
                },
                {
                "EmailAddress": "fieldnotindef5@test.elq",
                "notindeffield1": "1",
                "notindeffield2": "2",
                "notindeffield3": "3",
                "notindeffield4": "4",
                "notindeffield5": "5"
                }
        ]
```


Sync logs:


```
{
                "items": [
                {
                "syncUri": "/syncs/178",
```

```
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield2) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
```

```
                                    "severity": "warning",
                                    "statusCode": "ELQ-00111",
                                    "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
                                    "createdAt": "2017-02-16T22:39:48.5230000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/178",
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00111",
                                    "message": "Field (notindeffield2) is not part of import definition, ar
ignored.",
                                    "createdAt": "2017-02-16T22:39:48.5230000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/178",
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00111",
                                    "message": "Field (notindeffield3) is not part of import definition, ar
ignored.",
                                    "createdAt": "2017-02-16T22:39:48.5230000Z"
                                    },
                                    {
                                    "syncUri": "/syncs/178",
                                    "count": 0,
                                    "severity": "warning",
```

```
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield2) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield3) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
```

```
                                "message": "Field (notindeffield4) is not part of import definition, a
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield2) is not part of import definition, ar
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield3) is not part of import definition, ar
```

```
ignored.",
                                        "createdAt": "2017-02-16T22:39:48.5230000Z"
                                        },
                                        {
                                        "syncUri": "/syncs/178",
                                        "count": 0,
                                        "severity": "warning",
                                        "statusCode": "ELQ-00111",
                                        "message": "Field (notindeffield4) is not part of import definition, a
ignored.",
                                        "createdAt": "2017-02-16T22:39:48.5230000Z"
                                        },
                                        {
                                        "syncUri": "/syncs/178",
                                        "count": 0,
                                        "severity": "warning",
                                        "statusCode": "ELQ-00111",
                                        "message": "Field (notindeffield5) is not part of import definition, a
ignored.",
                                        "createdAt": "2017-02-16T22:39:48.5230000Z"
                                        },
                                        {
                                        "syncUri": "/syncs/178",
                                        "count": 5,
                                        "severity": "information",
                                        "statusCode": "ELQ-00130",
                                        "message": "Total records staged for import.",
                                        "createdAt": "2017-02-16T22:39:48.7430000Z"
```

```
},
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00137",
"message": "Ready for data import processing.",
"createdAt": "2017-02-16T22:39:48.7600000Z"
},
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00101",
"message": "Sync processed for sync , resulting in Warning status."
"createdAt": "2017-02-16T22:39:50.1500000Z"
},
{
"syncUri": "/syncs/178",
"count": 5,
"severity": "information",
"statusCode": "ELQ-00001",
"message": "Total records processed.",
"createdAt": "2017-02-16T22:39:48.7200000Z"
},
{
"syncUri": "/syncs/178",
"count": 5,
```

```
                "severity": "information",
                "statusCode": "ELQ-00004",
                "message": "Contacts created.",
                "createdAt": "2017-02-16T22:39:49.5430000Z"
                },
                {
                "syncUri": "/syncs/178",
                "count": 0,
                "severity": "information",
                "statusCode": "ELQ-00022",
                "message": "Contacts updated.",
                "createdAt": "2017-02-16T22:39:49.5430000Z"
                }
                ],
                "totalResults": 21,
                "limit": 1000,
                "offset": 0,
                "count": 21,
                "hasMore": false
        }
```


Example: Retrieving a sync's logs in Release 492

Contact import definition:


```
{
                "name": "test",
                "fields": {
                "EmailAddress": "{{Contact.Field(C_EmailAddress)}}"
                },
```

```
                "identifierFieldName": "EmailAddress",
                "syncActions": [],
                "isSyncTriggeredOnImport": false,
                "dataRetentionDuration": "P7D",
                "isUpdatingMultipleMatchedRecords": false,
                "uri": "/contacts/imports/47",
                "createdBy": "API.User",
                "createdAt": "2017-02-14T23:43:20.4570000Z",
                "updatedBy": "API.User",
                "updatedAt": "2017-02-14T23:43:20.4570000Z"
            }
```


Uploaded data:


```
[
                {
                "EmailAddress": "fieldnotindef1@test.elq",
                "notindeffield1": "1"
                },
                {
                "EmailAddress": "fieldnotindef2@test.elq",
                "notindeffield1": "1",
                "notindeffield2": "2"
                },
                {
                "EmailAddress": "fieldnotindef3@test.elq",
                "notindeffield1": "1",
                "notindeffield2": "2",
                "notindeffield3": "3"
```

```
                                    },
                                    {
                                    "EmailAddress": "fieldnotindef4@test.elq",
                                    "notindeffield1": "1",
                                    "notindeffield2": "2",
                                    "notindeffield3": "3",
                                    "notindeffield4": "4"
                                    },
                                    {
                                    "EmailAddress": "fieldnotindef5@test.elq",
                                    "notindeffield1": "1",
                                    "notindeffield2": "2",
                                    "notindeffield3": "3",
                                    "notindeffield4": "4",
                                    "notindeffield5": "5"
                                    }
                                ]
```


Sync logs:


```
{
                                    "items": [
                                    {
                                    "syncUri": "/syncs/178",
                                    "count": 0,
                                    "severity": "warning",
                                    "statusCode": "ELQ-00111",
                                    "message": "Field (notindeffield1) is not part of import definition, ar
ignored.",
```

```
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield2) is not part of import definition, a
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield3) is not part of import definition, a
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
                                {
                                "syncUri": "/syncs/178",
                                "count": 0,
                                "severity": "warning",
                                "statusCode": "ELQ-00111",
                                "message": "Field (notindeffield4) is not part of import definition, a
ignored.",
                                "createdAt": "2017-02-16T22:39:48.5230000Z"
                                },
```

```
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "warning",
"statusCode": "ELQ-00111",
"message": "Field (notindeffield5) is not part of import definition, a
ignored.",
"createdAt": "2017-02-16T22:39:48.5230000Z"
},
{
"syncUri": "/syncs/178",
"count": 5,
"severity": "information",
"statusCode": "ELQ-00130",
"message": "Total records staged for import.",
"createdAt": "2017-02-16T22:39:48.7430000Z"
},
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00137",
"message": "Ready for data import processing.",
"createdAt": "2017-02-16T22:39:48.7600000Z"
},
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "information",
```

```
"statusCode": "ELQ-00101",
"message": "Sync processed for sync , resulting in Warning status."
"createdAt": "2017-02-16T22:39:50.1500000Z"
},
{
"syncUri": "/syncs/178",
"count": 5,
"severity": "information",
"statusCode": "ELQ-00001",
"message": "Total records processed.",
"createdAt": "2017-02-16T22:39:48.7200000Z"
},
{
"syncUri": "/syncs/178",
"count": 5,
"severity": "information",
"statusCode": "ELQ-00004",
"message": "Contacts created.",
"createdAt": "2017-02-16T22:39:49.5430000Z"
},
{
"syncUri": "/syncs/178",
"count": 0,
"severity": "information",
"statusCode": "ELQ-00022",
"message": "Contacts updated.",
"createdAt": "2017-02-16T22:39:49.5430000Z"
}
],
```

```
        "totalResults": 11,
        "limit": 1000,
        "offset": 0,
        "count": 11,
        "hasMore": false
    }
```


Recent changes

Application API

- Prior to 492, it was possible to update the `runAsUserId` property while a campaign was Active using the Update a campaign endpoint. This could have resulted in users not being able to deactivate campaigns. With 492 the `runAsUserId` property will no longer be able to be changed while a campaign is Active.

Release 491

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 491.

New features

Application API

- We have modified the endpoints used to retrieve Events, so that partial depth will return all properties except for `sessions`, `sessionFieldValues`, `emailAddressFieldId`, `eventGroupByFieldId`, and `uniqueCodeFieldId`. Currently, partial depth returns all properties. Learn more

- You can now use the Contact Segments API in REST 2.0 to create, update, retrieve, delete, and copy contact segments. Learn more

Bulk API

- Added the ability to include up to 10 contact fields in an activity export definition. Note that the addition of contact fields to activity exports will add to export time. If more than 10 contact fields are included, Eloqua will respond with a 400 validation error. The syntax for a contact field statement in a Bulk API activity export definition is:

  `{{Activity.Contact.Field(<contact_field_name>)}}`

  Example

  Create an activity export definition using the contact field `ContactIdExt` to retrieve a contact's Eloqua ID:

  Request

  ```
  POST /api/bulk/2.0/activities/exports
  ```

  Request body

  ```
  {
                                  "filter": "'{{Activity.Type}}'='EmailOpen'",
                                  "name": "Bulk Activity Export - Email Open",
                                  "fields": {
                                  "ActivityId": "{{Activity.Id}}",
                                  "ActivityType": "{{Activity.Type}}",
                                  "ActivityDate": "{{Activity.CreatedAt}}",
                                  "ContactId": "{{Activity.Contact.Id}}",
                                  "IpAddress": "{{Activity.Field(IpAddress)}}",
                                  "VisitorId": "{{Activity.Visitor.Id}}",
  ```

```
                "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
                "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                "AssetType": "{{Activity.Asset.Type}}",
                "AssetName": "{{Activity.Asset.Name}}",
                "AssetId": "{{Activity.Asset.Id}}",
                "SubjectLine": "{{Activity.Field(SubjectLine)}}",
                "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
                "CampaignId": "{{Activity.Campaign.Id}}",
                "ExternalId": "{{Activity.ExternalId}}",
                "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                "EmailSendType": "{{Activity.Field(EmailSendType)}}",
                "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
                }
        }
```

Response body

```
{
                "name": "Bulk Activity Export - Email Open",
                "fields": {
                "ActivityId": "{{Activity.Id}}",
                "ActivityType": "{{Activity.Type}}",
                "ActivityDate": "{{Activity.CreatedAt}}",
                "ContactId": "{{Activity.Contact.Id}}",
                "IpAddress": "{{Activity.Field(IpAddress)}}",
                "VisitorId": "{{Activity.Visitor.Id}}",
                "VisitorExternalId": "{{Activity.Visitor.ExternalId}}",
```

```
                "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                "AssetType": "{{Activity.Asset.Type}}",
                "AssetName": "{{Activity.Asset.Name}}",
                "AssetId": "{{Activity.Asset.Id}}",
                "SubjectLine": "{{Activity.Field(SubjectLine)}}",
                "EmailWebLink": "{{Activity.Field(EmailWebLink)}}",
                "CampaignId": "{{Activity.Campaign.Id}}",
                "ExternalId": "{{Activity.ExternalId}}",
                "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                "EmailSendType": "{{Activity.Field(EmailSendType)}}",
                "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                "ContactIdExt": "{{Activity.Contact.Field(ContactIDExt)}}"
                },
                "filter": "'{{Activity.Type}}'='EmailOpen'",
                "dataRetentionDuration": "PT12H",
                "uri": "/activities/exports/1560",
                "createdBy": "API.User",
                "createdAt": "2017-08-09T16:22:09.0434041Z",
                "updatedBy": "API.User",
                "updatedAt": "2017-08-09T16:22:09.0434041Z"
            }
```

- You can now include the following fields when creating a Bounceback activity export definition:
    - `EmailRecipientId`
    - `EmailDeploymentId`
    - `SmtpErrorCode`

- `SmtpStatusCode`
- `SmtpMessage`

This enhancement enables developers to tie a hard Bounceback to an email send, and provides further details on the Bounceback. For more information on these fields, see Activity Fields.

Example

Create a Bounceback activity export definition:

Request

```
POST /api/bulk/2.0/activities/exports
```

Request body

```
{
                                    "filter": "'{{Activity.Type}}'='Bounceback'",
                                    "name": "Bulk Activity Export - Bounceback",
                                    "fields": {
                                    "ActivityId": "{{Activity.Id}}",
                                    "ActivityType": "{{Activity.Type}}",
                                    "ActivityDate": "{{Activity.CreatedAt}}",
                                    "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                                    "AssetType": "{{Activity.Asset.Type}}",
                                    "AssetName": "{{Activity.Asset.Name}}",
                                    "AssetId": "{{Activity.Asset.Id}}",
                                    "CampaignId": "{{Activity.Campaign.Id}}",
                                    "ExternalId": "{{Activity.ExternalId}}",
                                    "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                                    "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
```

```
                "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
                "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
                "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
                }
            }
```

Response body

```
{
                "name": "Bulk Activity Export - Bounceback",
                "fields": {
                "ActivityId": "{{Activity.Id}}",
                "ActivityType": "{{Activity.Type}}",
                "ActivityDate": "{{Activity.CreatedAt}}",
                "EmailAddress": "{{Activity.Field(EmailAddress)}}",
                "AssetType": "{{Activity.Asset.Type}}",
                "AssetName": "{{Activity.Asset.Name}}",
                "AssetId": "{{Activity.Asset.Id}}",
                "CampaignId": "{{Activity.Campaign.Id}}",
                "ExternalId": "{{Activity.ExternalId}}",
                "EmailRecipientId": "{{Activity.Field(EmailRecipientId)}}",
                "DeploymentId": "{{Activity.Field(EmailDeploymentId)}}",
                "SmtpErrorCode": "{{Activity.Field(SmtpErrorCode)}}",
                "SmtpStatusCode": "{{Activity.Field(SmtpStatusCode)}}",
                "SmtpMessage": "{{Activity.Field(SmtpMessage)}}"
                },
                "filter": "'{{Activity.Type}}'='Bounceback'",
                "dataRetentionDuration": "PT12H",
                "uri": "/activities/exports/51",
```

```
                "createdBy": "API.User",
                "createdAt": "2017-08-18T17:39:41.3427356Z",
                "updatedBy": "API.User",
                "updatedAt": "2017-08-18T17:39:41.3427356Z"
            }
```


Retrieve the data:

```
GET /api/bulk/2.0/syncs/26697/data?limit=1
```

Response body:


```
{
                "totalResults": 177417,
                "limit": 1,
                "offset": 0,
                "count": 1,
                "hasMore": true,
                "items": [
                {
                "ActivityId": "22838",
                "ActivityType": "Bounceback",
                "ActivityDate": "2015-08-21 02:47:07.000",
                "EmailAddress": "john.doe@gmail.com",
                "AssetType": "Email",
                "AssetName": "Modern Marketing Experience 2018",
                "AssetId": "2842",
                "CampaignId": "1482",
                "ExternalId": "BBM1P3000000022838",
```

```
                                        "EmailRecipientId": "08daac6a-5de0-47ce-b99e-b73f05c97ff9",
                                        "DeploymentId": "4197",
                                        "SmtpErrorCode": "5.1.1",
                                        "SmtpStatusCode": "550",
                                        "SmtpMessage": "The email account that you tried to reach does no
Please try\\r\\n550-5.1.1 double-checking the recipient's email address for typos
or\\r\\n550-5.1.1 unnecessary spaces. Learn more at\\r\\n550 5.1.1
https://support.google.com/mail/answer/6596"
                                    },
```


Recent changes

Bulk API

- Resolved an instance where creating a Feeder sync action definition for custom objects was causing an error.
- Modified the behavior of the `ExternalId` field on custom objects. The `ExternalId` field can now be used as the `identifierFieldName` on bulk imports even if there is a unique code field set on the destination custom object.

Release 490

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 490.

New features

Application API

- When retrieving a list of custom object data in REST 2.0, you can now filter by `uniqueCode` as a `search` query parameter to filter results. Each custom object data entity contains a unique code that can be used in a filter. Learn more

  Example

  In REST 2.0, retrieve a list of custom object data and filter results by unique code:

  Request

  ```
  GET
  /api/rest/2.0/data/customObject/25/instances?search=uniquecode=DDC13000000000002
  ```

  Response

  ```
  {
                                  "elements": [
                                  {
                                  "type": "CustomObjectData",
                                  "id": "2",
                                  "createdAt": "1477423778",
                                  "name": "(2)",
                                  "updatedAt": "1477423778",
                                  "fieldValues": [
                                  {
                                  "type": "FieldValue",
                                  "id": "62",
                                  "value": "Completed"
  ```

```
},
{
"type": "FieldValue",
"id": "61",
"value": "Completed"
},
{
"type": "FieldValue",
"id": "60",
"value": "Completed"
},
{
"type": "FieldValue",
"id": "63",
"value": "jon.doe@oracle.com"
},
{
"type": "FieldValue",
"id": "64",
"value": "Jon"
},
{
"type": "FieldValue",
"id": "65",
"value": "Doe"
},
{
"type": "FieldValue",
"id": "66",
```

```
                "value": "M5A4C9"
                }
                ],
                "uniqueCode": "DDC130000000000002"
                }
                ],
                "page": 1,
                "pageSize": 1000,
                "total": 1
        }
```


Recent changes

Application API

- Resolved an issue where retrieving a list of custom object data in REST 2.0 when ordering by `updatedAt` would result in an error. You can now send this request ordering results using `updatedAt` in REST 2.0.

  Example

  Retrieve a list of custom object data and order results by updated date:

  Request

  ```
  GET /api/rest/2.0/data/customObject/25/instances?orderby=updatedAt
  ```

- The custom object data API in REST 1.0 and 2.0 will no longer result in an error when using `name` in the search query parameter to filter results.

  Examples

  In REST 1.0, retrieve a list of custom object data and filter results by name:

Request

```
GET /api/rest/1.0/data/customObject/25?search=name=john.doe@oracle.com
```

In REST 2.0, retrieve a list of custom object data and filter results by name:

Request

```
GET
/api/rest/2.0/data/customObject/25instances?search=name=john.doe@oracle.com
```

Release 489

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 489.

New features

Bulk API

Syncs

- We've created a new endpoint to allow developers to delete sync data from the staging area using sync `id`:

  `DELETE /api/bulk/2.0/syncs/{id}/data`. Learn more

  This new endpoint enables developers to delete sync data from the staging area, without having to use the various definitions data endpoints available. Prior to Release 489, to delete sync data, developers needed to use the specific definition data endpoint (`/<entity>/exports/{id}/data`). This method removes data for any syncs using the definition.

The new `DELETE /api/bulk/2.0/syncs/{id}/data` endpoint provides developers another way to delete sync data using sync `id`, without removing data for the same definition within a different sync.

Examples: Delete sync data

Delete sync data using sync id

Delete the data for the sync with ID = 40 (use this endpoint to delete sync data tied to a specific sync):

Request

```
DELETE /api/bulk/2.0/syncs/40/data
```

Response status (For `DELETE` requests, no body is returned):

```
204 No Content
```

Delete sync data using definition data endpoint

Delete the data for the contact export definition with ID = 50 (use this endpoint to delete all sync data tied to a definition):

Request

```
DELETE /api/bulk/2.0/contacts/exports/50/data
```

Response status (For `DELETE` requests, no body is returned)**:**

```
204 No Content
```

Recent changes

Application API

- When retrieving campaigns, we've modified the properties returned when `depth` is set to `partial`. Prior to Release 489, partial depth would return all properties. Partial depth will now return all properties except for:
  - `elements`
  - `isReadOnly`
  - `isMemberAllowedReEntry`

  Learn more

- The form data API `api/REST/1.0/data/form` will no longer create a new visitor or new thread when no field with the HTML name `elqCustomerGUID` is passed to the REST API. As no thread is being created, form submits will not create a new website visit activity record.

Release 488

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 488.

New features

Application API

Events

- You can now use the events API endpoints to create, update, retrieve, and delete events. This functionality was previously only supported via the SOAP API, which has been deprecated. These new endpoints provide access to event fields that were not available via the SOAP API. Learn more

Example: Create an event

Create an event:

```
POST api/REST/2.0/assets/eventRegistration
                                    Content-Type: application/json
```

Request body:

```
{
                                    "name": "Modern Marketing Experience",
                                    "description": "Modern Marketing Experience brings together Mode
from around the globe",
                                    "emailAddressFieldId": "-2",
                                    "fields": [
                                    {
                                    "type": "CustomObjectField",
                                    "id": "-2",
                                    "name": "Email",
                                    "dataType": "text",
                                    "displayType": "text"
                                    },
                                    {
                                    "type": "CustomObjectField",
                                    "name": "First Name",
                                    "dataType": "text",
                                    "displayType": "text"
                                    },
                                    {
                                    "type": "CustomObjectField",
```

```
"name": "Industry",
"dataType": "text",
"displayType": "singleSelect",
"optionListId": "9"
},
{
"type": "CustomObjectField",
"name": "Markie Submitted",
"checkedValue": "Yes",
"dataType": "text",
"defaultValue": "No",
"displayType": "checkbox",
"uncheckedValue": "No"
},
{
"type": "CustomObjectField",
"name": "Birthday",
"dataType": "date",
"displayType": "text"
},
{
"type": "CustomObjectField",
"name": "Years Using Eloqua",
"dataType": "number",
"displayType": "text"
},
{
"type": "CustomObjectField",
"name": "MME Location",
```

```
"dataType": "text",
"defaultValue": "North America",
"displayType": "singleSelect",
"optionListId": "108"
}
],
"sessionFields": [
{
"type": "EventSessionField",
"name": "Date",
"dataType": "date",
"outputFormat": {
"type": "FieldOutputFormat",
"id": "19"
}
},
{
"type": "EventSessionField",
"name": "Location",
"dataType": "text"
}
],
"sessions": [
{
"type": "EventSession",
"name": "North America"
},
{
"type": "EventSession",
```

```
                "name": "Europe"
                }
                ],
                "uniqueCodeFieldId": "-2"
            }
```


Request response:


```
{
                "type": "EventRegistration",
                "id": "1260",
                "createdAt": "1493156690",
                "createdBy": "71",
                "depth": "complete",
                "description": "Modern Marketing Experience brings together Mode
from around the globe",
                "folderId": "81",
                "name": "Modern Marketing Experience",
                "updatedAt": "1493156690",
                "updatedBy": "71",
                "emailAddressFieldId": "3958",
                "fields": [
                {
                "type": "CustomObjectField",
                "id": "3958",
                "initialId": "-2",
                "depth": "complete",
                "name": "Email",
                "dataType": "text",
```

```
"displayType": "text",
"internalName": "Email1"
},
{
"type": "CustomObjectField",
"id": "3959",
"depth": "complete",
"name": "First Name",
"dataType": "text",
"displayType": "text",
"internalName": "First_Name1"
},
{
"type": "CustomObjectField",
"id": "3960",
"depth": "complete",
"name": "Industry",
"dataType": "text",
"displayType": "singleSelect",
"internalName": "Industry1",
"optionListId": "9"
},
{
"type": "CustomObjectField",
"id": "3961",
"depth": "complete",
"name": "Markie Submitted",
"checkedValue": "Yes",
"dataType": "text",
```

```
"defaultValue": "No",
"displayType": "checkbox",
"internalName": "Markie_Submitted1",
"uncheckedValue": "No"
},
{
"type": "CustomObjectField",
"id": "3962",
"depth": "complete",
"name": "Birthday",
"dataType": "date",
"displayType": "text",
"internalName": "Birthday1"
},
{
"type": "CustomObjectField",
"id": "3963",
"depth": "complete",
"name": "Years Using Eloqua",
"dataType": "number",
"displayType": "text",
"internalName": "Years_Using_Eloqua1"
},
{
"type": "CustomObjectField",
"id": "3964",
"depth": "complete",
"name": "MME Location",
"dataType": "text",
```

```
"defaultValue": "North America",
"displayType": "singleSelect",
"internalName": "MME_Location1",
"optionListId": "108"
}
],
"recordCount": 0,
"sessionFields": [
{
"type": "EventSessionField",
"id": "535",
"name": "Date",
"dataType": "date",
"outputFormat": {
"type": "FieldOutputFormat",
"id": "19",
"dataType": "date",
"format": "dddd MMMM d, yyyy"
}
},
{
"type": "EventSessionField",
"id": "536",
"name": "Location",
"dataType": "text"
}
],
"sessions": [
{
```

```
                "type": "EventSession",
                "id": "51",
                "name": "North America"
                },
                {
                "type": "EventSession",
                "id": "52",
                "name": "Europe"
                }
                ],
                "uniqueCodeFieldId": "3958"
        }
```

Event Registrants

- You can now use the Event registrants API endpoints to create, update, retrieve, and delete event registrants. This provides a low volume synchronous option to go along with the Bulk API endpoints to be used for large volumes. Learn more

  Example: Create an event registrant

  Create a new event registrant and map to contact for the event with Id #1260:

```
POST api/REST/2.0/data/eventRegistration/1260/instance
                Content-Type: application/json
```

  Request body:

```
{
                "contactId": "1653175",
                "fieldValues": [
```

```
{
"id": "3958",
"value": "sally@oracle.com"
},
{
"id": "3959",
"value": "Sally"
},
{
"id": "3960",
"value": "Technology"
},
{
"id": "3962",
"value": "41547123"
},
{
"id": "3964",
"value": "North America"
},
{
"id": "3963",
"value": "2"
},
{
"id": "3961",
"value": "Yes"
}
]
```

```
                                    }
```

Request response:

```
{
                                        "type": "EventRegistrationData",
                                        "id": "332379",
                                        "createdAt": "1493240064",
                                        "name": "sally@oracle.com",
                                        "updatedAt": "1493240064",
                                        "contactId": "1653175",
                                        "customObjectRecordStatus": "Registered",
                                        "fieldValues": [
                                        {
                                        "type": "FieldValue",
                                        "id": "3958",
                                        "value": "sally@oracle.com"
                                        },
                                        {
                                        "type": "FieldValue",
                                        "id": "3959",
                                        "value": "Sally"
                                        },
                                        {
                                        "type": "FieldValue",
                                        "id": "3960",
                                        "value": "Technology"
                                        },
                                        {
```

```
                "type": "FieldValue",
                "id": "3962",
                "value": "41547123"
                },
                {
                "type": "FieldValue",
                "id": "3964",
                "value": "North America"
                },
                {
                "type": "FieldValue",
                "id": "3963",
                "value": "2"
                },
                {
                "type": "FieldValue",
                "id": "3961",
                "value": "Yes"
                }
                ],
                "isMapped": "Yes",
                "uniqueCode": "sally@oracle.com"
        }
```


Platform notices

- When importing invalid data along with valid data that is imported, the sync logs will now indicate the record in the count for statusCode `ELQ-00144` that has a message of "Total records with rejected fields." Learn more

Release 487

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 487.

New features

AppCloud Developer Framework

- Added the ability to retrieve a low-volume batch of service instances through the AppCloud API. App providers can now retrieve up to 200 service instances in a single call for Action and Decision services. Note that `X-HTTP-Method-Override: SEARCH` is required in the request header. See the endpoint documentation for Action and Decision services for more information.

  Example: Retrieve 2 action service instances

  Retrieve 2 action service instances:

  ```
  POST api/cloud/1.0/actions/instances
                                      X-HTTP-Method-Override: SEARCH
                                  Content-Type: application/json
  ```

  Request body:

  ```
  {
                                      "ids": [
                                      "d7a34c94-c370-4958-8a88-b56d5621e68a",
                                      "1869b464-56f2-4334-923a-e631849d3cc8"
                                      ]
                                  }
  ```

  Response:

```
{
                                    "items": [
                                    {
                                    "recordDefinition": {
                                    "Id": "{{Contact.Id}}",
                                    "field2": "{{Contact.Field(C_FirstName)}}"
                                    },
                                    "requiresConfiguration": false,
                                    "statement": "{{ActionInstance(d7a34c94c37049588a88b56d5621e
                                    "dependentAssetType": "Campaign",
                                    "dependentAssetId": 49251,
                                    "uri": "/actions/618068/instances/d7a34c94c37049588a88b56d5
                                    "createdBy": "Admin.User",
                                    "createdAt": "2017-03-03T15:31:32.8930000Z",
                                    "updatedBy": "Admin.User",
                                    "updatedAt": "2017-03-03T15:31:33.4000000Z"
                                    },
                                    {
                                    "recordDefinition": {
                                    "Email": "{{CustomObject[1].Field[4]}}"
                                    },
                                    "requiresConfiguration": false,
                                    "statement": "{{ActionInstance(1869b46456f24334923ae631849d3
                                    "dependentAssetType": "Program",
                                    "dependentAssetId": 49252,
                                    "uri": "/actions/618068/instances/1869b46456f24334923ae63184
                                    "createdBy": "Admin.User",
                                    "createdAt": "2017-03-03T15:31:33.8930000Z",
                                    "updatedBy": "Admin.User",
```

```
                "updatedAt": "2017-03-03T15:31:34.2170000Z"
                }
                ]
        }
```

Release 486

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 486.

New features

AppCloud Developer Framework

- Firehose services can now subscribe to program canvas events. This enhancement provides App providers the ability to develop firehose services that receive notifications for the following program canvas events: Created, Updated, Deleted, Draft, Activated, and Paused. Learn more

  Example: Firehose service configured to request all program-related events

  For example, if an app has a Firehose service and is configured to request all program-related events. A call to its notification URL could then resemble:

```
POST https://example.com/firehose

                {
                "siteId": 12345678,
                "assetId": 1234,
                "assetType": "Program",
```

```
                "eventType": "Created",
                "eventDate": "02/09/2017 10:27:51",
                "userId": "2",
                "userName": "API.User",
                "siteName": "API.Test",
                "msgAttributes": {
                "name": "Example Program",
                "userculture": "en-US"
                }
        }
```


Platform notices

- Support is ending for the EloquaService SOAP API on April 1st, 2017. Learn more

Release 485

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 485.

New features

AppCloud Developer Framework

- Increased the canvas cloud step data retention from 30 days to 90 days.

Events

- We've added new endpoints to enable importing event registrants. These endpoints were built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated. Learn more.

  Example: Create an import definition for an event

  Create a new event (id #12) import:

```
POST api/bulk/2.0/events/12/imports
                                    Content-Type: application/json
```

  Request body:

```
{
                                        "name": "Winter 2016 Fall Fashion Show Event Import",
                                        "fields": {
                                        "first_name": "{{Event[12].Field[55]}}",
                                        "last_name": "{{Event[12].Field[56]}}",
                                        "email": "{{Event[12].Field[57]}}"
                                        },
                                        "identifierFieldName": "email",
                                        "isSyncTriggeredOnImport" : "false"
                                    }
```

  Request response:

```
{
                                        "id": 12,
```

```
                "parentId": 12,
                "mapDataCardsCaseSensitiveMatch": false,
                "name": "Winter 2016 Fall Fashion Show Event Import",
                "fields": {
                "first_name": "{{Event[12].Field[55]}}",
                "last_name": "{{Event[12].Field[56]}}",
                "email": "{{Event[12].Field[57]}}"
                },
                "identifierFieldName": "email",
                "isSyncTriggeredOnImport": false,
                "dataRetentionDuration": "P7D",
                "isUpdatingMultipleMatchedRecords": false,
                "uri": "/events/12/imports/12",
                "createdBy": "API.User",
                "createdAt": "2016-11-24T15:54:26.6704371Z",
                "updatedBy": "API.User",
                "updatedAt": "2016-11-24T15:54:26.6704371Z"
        }
```


Recent changes

Authentication

- Improved the behaviour of OAuth when simultaneous requests are sent to refresh an access token.

Release 484

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 484.

New features

AppCloud Developer Framework

- The program canvas is now generally available, and app developers can now specify if their service is available for use in contact programs and custom object programs (during service registration, under **User Access**). We've also added two new service level URL template parameters for action, decision, and feeder services. The `EntityType` and `CustomObjectId` parameters support developers wanting to create apps for the program canvas. Learn more

Bulk API

Events

- We've added new endpoints that enable developers to discover Events and Event Fields. Learn more

  Example: Retrieving events in your database

  Retrieve the first 2 events in your database:

  ```
  GET api/bulk/2.0/events?limit=2
  ```

  Request response:

  ```
  {
                                  "items": [
                                  {
                                  "name": "Example Event",
                                  "displayNameFieldUri": "/events/5/fields/34",
                                  "emailAddressFieldUri": "/events/5/fields/34",
                                  "uniqueFieldUri": "/events/5/fields/34",
  ```

```
                "createdBy": "API.User",
                "createdAt": "2016-03-23T15:04:21.8400000Z",
                "updatedBy": "API.User",
                "updatedAt": "2016-05-30T18:00:46.9200000Z",
                "uri": "/events/5"
                },
                {
                "name": "Summer 2016 Hotdog Eating Contest",
                "createdBy": "API.User",
                "createdAt": "2016-10-21T20:51:56.8470000Z",
                "updatedBy": "API.User",
                "updatedAt": "2016-10-21T21:03:44.8500000Z",
                "uri": "/events/12"
                }
                ],
                "totalResults": 5,
                "limit": 2,
                "offset": 0,
                "count": 2,
                "hasMore": true
            }
```


Example: Retrieving fields for an event

Retrieve a list of the first 5 event fields for the event with id # 8:

```
GET api/bulk/2.0/events/8/fields?limit=5
```

Request response:

```
{
                                    "items": [
                                    {
                                    "name": "First Name",
                                    "internalName": "First_Name1",
                                    "dataType": "string",
                                    "defaultValue": "Name",
                                    "hasReadOnlyConstraint": false,
                                    "hasNotNullConstraint": false,
                                    "hasUniquenessConstraint": false,
                                    "statement": "{{Event[12].Field[55]}}",
                                    "uri": "/events/12/fields/55"
                                    },
                                    {
                                    "name": "Last Name",
                                    "internalName": "Last_Name1",
                                    "dataType": "string",
                                    "defaultValue": "Last Name",
                                    "hasReadOnlyConstraint": false,
                                    "hasNotNullConstraint": false,
                                    "hasUniquenessConstraint": false,
                                    "statement": "{{Event[12].Field[56]}}",
                                    "uri": "/events/12/fields/56"
                                    },
                                    {
                                    "name": "Email Address",
                                    "internalName": "Email_Address1",
                                    "dataType": "string",
```

```
"defaultValue": "Email Address",
"hasReadOnlyConstraint": false,
"hasNotNullConstraint": false,
"hasUniquenessConstraint": false,
"statement": "{{Event[12].Field[57]}}",
"uri": "/events/12/fields/57"
},
{
"name": "Job Role",
"internalName": "Job_Role1",
"dataType": "string",
"defaultValue": "Job Role",
"hasReadOnlyConstraint": false,
"hasNotNullConstraint": false,
"hasUniquenessConstraint": false,
"statement": "{{Event[12].Field[58]}}",
"uri": "/events/12/fields/58"
},
{
"name": "Industry",
"internalName": "Industry1",
"dataType": "string",
"defaultValue": "Industry",
"hasReadOnlyConstraint": false,
"hasNotNullConstraint": false,
"hasUniquenessConstraint": false,
"statement": "{{Event[12].Field[59]}}",
"uri": "/events/12/fields/59"
```

```
                    }
                    ],
                    "totalResults": 8,
                    "limit": 5,
                    "offset": 0,
                    "count": 5,
                    "hasMore": true
                }
```

- We've added new endpoints to enable exporting event registrants. These endpoints were built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated. Learn more.

  Example: Create an export definition for an event

  Create a new event (id #5) export:

```
POST api/bulk/2.0/events/5/exports
                    Content-Type: application/json
```

  Request body:

```
{
                    "name": "2016 Fall Fashion Show Event Export",
                    "fields": {
                    "Email": "{{Event[5].Field[33]}}",
                    "ID": "{{Event[5].Field[33]}}"
                    }
                }
```

Request response:

```
{
                                    "name": "2016 Fall Fashion Show Event Export",
                                    "fields": {
                                    "Email": "{{Event[5].Field[33]}}",
                                    "ID": "{{Event[5].Field[33]}}"
                                    },
                                    "dataRetentionDuration": "PT12H",
                                    "uri": "/events/5/exports/14",
                                    "createdBy": "API.User",
                                    "createdAt": "2016-10-27T16:14:16.9561996Z",
                                    "updatedBy": "API.User",
                                    "updatedAt": "2016-10-27T16:14:16.9561996Z"
                                }
```

- Endpoints to enable importing event registrants will be available in release 485.

Exports

- You can now use the `areSystemTimestampsInUTC` request parameter to export system timestamp fields into Coordinated Universal Time (UTC). By default, system timestamp fields are expressed in Eastern Time (ET). By setting `areSystemTimestampsInUTC` to `true`, you can export system timestamp fields in UTC on contacts, custom objects, accounts, activity records, and events.

  Example: Export accounts when areSystemTimestampsInUTC is set to true

  With the `areSystemTimestampsInUTC` request parameter set to `true`, export accounts that have been created after September 9, 2016 13:46:20.975 UTC:

  Example account export definition filter:

```
{
                                    "name": "System Timestamps in UTC Export",
                                    "fields": {
                                    "companyName": "{{Account.Field(M_CompanyName)}}",
                                    "createdDate": "{{Account.Field(M_DateCreated)}}",
                                    "modifiedDate": "{{Account.Field(M_DateModified)}}"
                                    },
                                    "areSystemTimestampsInUTC": true,
                                    "filter": "'{{Account.Field(M_DateCreated)}}' > '2016-09-01T13:46:20
                                }
```


After syncing and retrieving the data, here's the exported data.


```
{
                                    "totalResults": 4,
                                    "limit": 1000,
                                    "offset": 0,
                                    "count": 4,
                                    "hasMore": false,
                                    "items": [
                                    {
                                    "companyName": "ABC Industries",
                                    "createdDate": "2016-09-01T13:46:20.980Z",
                                    "modifiedDate": "2016-09-01T13:46:20.980Z"
                                    },
                                    {
                                    "companyName": "BCD Company",
                                    "createdDate": "2016-10-10T07:48:28.247Z",
                                    "modifiedDate": "2016-10-10T07:48:28.247Z"
```

```
        },
        {
        "companyName": "XYZ Inc.",
        "createdDate": "2016-10-10T07:49:02.737Z",
        "modifiedDate": "2016-10-10T07:49:40.713Z"
        },
        {
        "companyName": "Acne Inc.",
        "createdDate": "2016-10-10T11:55:56.150Z",
        "modifiedDate": "2016-10-10T11:55:56.150Z"
        }
        ]
}
```

Recent changes

Application API

- When retrieving campaigns, the `memberCount` property will now only appear if the request `depth` is set to `partial` or `complete`. Previously, `memberCount` was included if the request `depth` was set to `minimal`, `partial`, or `complete`. Learn more

- Eloqua's APIs will now correctly return a value of `true` for `hasNotNullConstraint` (Bulk API) and `isRequired` (Application API) in Eloqua instances where email addresses are required for contacts. Previously when retrieving the email address field, Eloqua's APIs would incorrectly indicate that email address field was not required by returning a value of `false` for `hasNotNullConstraint` in the Bulk API and `isRequired` in the Application API.

  Example: Retrieving a contact field via the Bulk API prior to release 484 in an instance where email address is required

  Retrieve the email address contact field:

```
GET /api/bulk/2.0/contacts/fields?q='internalName=C_EmailAddress'
```

Request response:

```
{
                                    "items": [
                                    {
                                    "name": "Email Address",
                                    "internalName": "C_EmailAddress",
                                    "dataType": "emailAddress",
                                    "hasReadOnlyConstraint": false,
                                    "hasNotNullConstraint": false,
                                    "hasUniquenessConstraint": true,
                                    "statement": "{{Contact.Field(C_EmailAddress)}}",
                                    "uri": "/contacts/fields/100001",
                                    "createdAt": "1900-01-01T05:00:00.0000000Z",
                                    "updatedAt": "1900-01-01T05:00:00.0000000Z"
                                    }
                                    ],
                                    "totalResults": 1,
                                    "limit": 1000,
                                    "offset": 0,
                                    "count": 1,
                                    "hasMore": false
                                }
```

Example: Retrieving a contact field via the Bulk API after 484 in an instance where email address is required

Retrieve the email address contact field:

```
GET /api/bulk/2.0/contacts/fields?q='internalName=C_EmailAddress'
```

Request response:


```
{
                                    "items": [
                                    {
                                    "name": "Email Address",
                                    "internalName": "C_EmailAddress",
                                    "dataType": "emailAddress",
                                    "hasReadOnlyConstraint": false,
                                    "hasNotNullConstraint": true,
                                    "hasUniquenessConstraint": true,
                                    "statement": "{{Contact.Field(C_EmailAddress)}}",
                                    "uri": "/contacts/fields/100001",
                                    "createdAt": "1900-01-01T05:00:00.0000000Z",
                                    "updatedAt": "1900-01-01T05:00:00.0000000Z"
                                    }
                                    ],
                                    "totalResults": 1,
                                    "limit": 1000,
                                    "offset": 0,
                                    "count": 1,
                                    "hasMore": false
                                }
```


Platform notices

- The Oracle Eloqua URL standardization that was to not renew the old certificates for secure.eloqua.com for POD 1 and www.02.secure.eloqua.com for POD 2 has been postponed

indefinitely.

The certificates in question will be renewed so that traffic from the URLs with "legacy" naming conventions will continue to be redirected appropriately and there will be no disruption on your end. **As there is no longer an expiration date, any calls (API calls, redirects, etc) to the old URLs will continue to work**.

Learn more

Release 483

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 483.

New features

Application API

Accounts

- We've added a new endpoint which retrieves account information for up to 200 accounts per request, searched by account id. Developers can now search on a low volume of accounts to retrieve their account information. This endpoint was built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated. Note that `X-HTTP-Method-Override: SEARCH` is required in the request header, and the `depth` parameter must be set in the response body instead of as a query string parameter. Learn more

  Example: Retrieve 2 accounts at minimal depth

  Retrieve 2 accounts at minimal depth:

```
POST api/REST/2.0/data/accounts
                                    X-HTTP-Method-Override: SEARCH
                                Content-Type: application/json
```

Request body:

```
{
                                    "ids": [2, 3],
                                    "depth": "minimal"
                                }
```

Request response:

```
[
                                    {
                                    "type": "Account",
                                    "id": "2",
                                    "createdAt": "1470257701",
                                    "depth": "minimal",
                                    "description": "This is an example account",
                                    "name": "Cyberdyne Systems",
                                    "updatedAt": "1470257701",
                                    "address1": "123 Industry Lane",
                                    "address2": "567 Company Road",
                                    "address3": "1738 Market Circle",
                                    "businessPhone": "(888) 757-6662",
                                    "city": "Los Angeles",
                                    "fieldValues": [
```

```
{
"type": "FieldValue",
"id": "100102"
},
{
"type": "FieldValue",
"id": "100094",
"value": "ME10T000000000002"
}
],
"postalCode": "45317"
},
{
"type": "Account",
"id": "3",
"createdAt": "1470257701",
"depth": "minimal",
"description": "",
"name": "Umbrella Corporation",
"updatedAt": "1470257701",
"address1": "88 Capcom Blvd.",
"address2": "145 Biomede Lane",
"address3": "1738 Resident Drive",
"businessPhone": "(303) 455-3930",
"city": "Riverside",
"fieldValues": [
{
"type": "FieldValue",
```

```
                "id": "100102"
                },
                {
                "type": "FieldValue",
                "id": "100094",
                "value": "ME10T000000000003"
                }
                ],
                "postalCode": "45318"
                }
        ]
```


- We've added a new endpoint to retrieve a list of all of the account groups that an account is a member of. Learn more

  Example: Retrieve a list of the account groups that an account is a member of, when the account id = 1:

  ```
  GET api/REST/2.0/data/account/1/membership
  ```

  Request response:


```
[
                {
                "type": "AccountGroup",
                "id": "1",
                "name": "Interscope Group"
                },
                {
```

```
            "type": "AccountGroup",
            "id": "2",
            "name": "Waterfront Group"
            },
            {
            "type": "AccountGroup",
            "id": "3",
            "name": "Hudsucker Group"
            },
            {
            "type": "AccountGroup",
            "id": "4",
            "name": "Vami Industries"
            }
    ]
```

Account groups

- You can now use the account groups API to create, update, retrieve, and delete account groups. Learn more

Contacts

- We've added a new endpoint to retrieve contact information for up to 200 contacts per request, searched by contact `id`. Developers can now search on a low volume of contacts to retrieve their contact information. This endpoint was built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated. Note that `X-HTTP-Method-Override: SEARCH` is required in the request header, and the `depth` parameter must be set in the response body instead of as a query string parameter. Learn more

  Example: Retrieve 2 contacts at minimal depth

```
POST api/REST/2.0/data/contacts
                                    X-HTTP-Method-Override: SEARCH
                                Content-Type: application/json
```

Request body:

```
{
                                    "ids": [1, 2],
                                    "depth": "minimal"
                                }
```

Request response:

```
[
                                    {
                                    "type": "Contact",
                                    "id": "1",
                                    "createdAt": "1418667629",
                                    "depth": "minimal",
                                    "name": "api.user@test.oracle.com",
                                    "updatedAt": "1466689992"
                                    },
                                    {
                                    "type": "Contact",
                                    "id": "2",
                                    "createdAt": "1418673492",
                                    "depth": "minimal",
                                    "name": "api.user2@test.oracle.com",
```

```
                "updatedAt": "1469552212"
                }
        ]
```

- We've added a new endpoint to retrieve a list of all of the contact lists that a contact is a member of. Learn more

  Example: Retrieve a list of the contact lists that a contact is a member of, when the contact id = 1:

```
GET api/REST/2.0/data/contact/1/membership
```

Request response:

```
[
                {
                "type":"ContactList",
                "id":"44",
                "name":"Summer 2016 Event Registrants"
                },
                {
                "type":"ContactList",
                "id":"53",
                "name":"Poutine Contest Attendees"
                },
                {
                "type":"ContactList",
                "id":"55",
                "name":"Summer 2016 Hotdog Eating Competition Winners"
```

```
        },
        {
        "type":"ContactList",
        "id":"57",
        "name":"Summer 2016 CNE Attendees"
        },
        {
        "type":"ContactList",
        "id":"116",
        "name":"Soft Bouncebacks"
        }
    ]
```

Signature rules

- You can now use the Signature Rules API to retrieve a single signature rule asset or a list of signature rules. Learn more

Visitor profiles

- We've added a new endpoint to retrieve visitor profile fields. You can now retrieve the fields and their field values associated to your website visitors. This was built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated. Learn more

  Example: Retrieve a list of all of the visitor profile fields in your database

  ```
  GET api/REST/2.0/assets/visitor/fields
  ```

  Request response:

```
{
                                        "elements": [{
                                        "type": "ProfileField",
                                        "id": "264",
                                        "name": "Customer Guid",
                                        "dataType": "text",
                                        "internalName": "CustomerGuid",
                                        "length": "50"
                                        }, {
                                        "type": "ProfileField",
                                        "id": "157",
                                        "name": "Browser",
                                        "dataType": "text",
                                        "internalName": "V_Browser_Type",
                                        "length": "50"
                                        }, {
                                        "type": "ProfileField",
                                        "id": "256",
                                        "name": "City (from IP)",
                                        "dataType": "text",
                                        "internalName": "V_CityFromIP",
                                        "length": "50"
                                        }, {
                                        "type": "ProfileField",
                                        "id": "210",
                                        "name": "Company DNS Name",
                                        "dataType": "text",
                                        "internalName": "V_CompanyDNSName",
                                        "length": "50"
```

```
}, {
"type": "ProfileField",
"id": "246",
"name": "Company (from IP)",
"dataType": "largeText",
"internalName": "V_CompanyNameFromIP1",
"length": "200"
}, {
"type": "ProfileField",
"id": "245",
"name": "Country (from IP)",
"dataType": "text",
"internalName": "V_CountryFromIP",
"length": "50"
}, {
"type": "ProfileField",
"id": "241",
"name": "Country (from DNS)",
"dataType": "text",
"internalName": "V_CountryName",
"length": "50"
}, {
"type": "ProfileField",
"id": "169",
"name": "Current Total Pages",
"dataType": "number",
"internalName": "V_Current_Total_Pages",
"length": "4"
}, {
```

```
"type": "ProfileField",
"id": "168",
"name": "Current Visit Length",
"dataType": "number",
"internalName": "V_Current_Visit_Length",
"length": "4"
}, {
"type": "ProfileField",
"id": "217",
"name": "First Page In Visit",
"dataType": "largeText",
"internalName": "V_FirstPageInVisit",
"length": "200"
}, {
"type": "ProfileField",
"id": "212",
"name": "First Visit Date and Time",
"dataType": "date",
"internalName": "V_FirstVisitDateAndTime",
"length": "8"
}, {
"type": "ProfileField",
"id": "160",
"name": "Internet Site",
"dataType": "text",
"internalName": "V_HostName",
"length": "50"
}, {
"type": "ProfileField",
```

```
"id": "162",
"name": "IPAddress",
"dataType": "text",
"internalName": "V_IPAddress",
"length": "50"
}, {
"type": "ProfileField",
"id": "254",
"name": "ISP (from IP)",
"dataType": "largeText",
"internalName": "V_ISPFromIP",
"length": "200"
}, {
"type": "ProfileField",
"id": "231",
"name": "Last Page In Visit",
"dataType": "text",
"internalName": "V_LastPageInVisit",
"length": "50"
}, {
"type": "ProfileField",
"id": "232",
"name": "Last Visit Date and Time",
"dataType": "date",
"internalName": "V_LastVisitDateAndTime",
"length": "8"
}, {
"type": "ProfileField",
"id": "258",
```

```
"name": "Latitude (from IP)",
"dataType": "text",
"internalName": "V_LatitudeFromIP",
"length": "50"
}, {
"type": "ProfileField",
"id": "259",
"name": "Longitude (from IP)",
"dataType": "text",
"internalName": "V_LongitudeFromIP",
"length": "50"
}, {
"type": "ProfileField",
"id": "167",
"name": "Name",
"dataType": "text",
"internalName": "V_Name",
"length": "50"
}, {
"type": "ProfileField",
"id": "255",
"name": "State/Province (from IP)",
"dataType": "text",
"internalName": "V_ProvinceFromIP",
"length": "50"
}, {
"type": "ProfileField",
"id": "240",
"name": "Time Zone",
```

```
"dataType": "text",
"internalName": "V_TimeZone",
"length": "50"
}, {
"type": "ProfileField",
"id": "239",
"name": "Time Zone Offset from GMT",
"dataType": "number",
"internalName": "V_TimeZoneOffsetMin",
"length": "4"
}, {
"type": "ProfileField",
"id": "164",
"name": "Total Pages",
"dataType": "number",
"internalName": "V_Total_Pages",
"length": "4"
}, {
"type": "ProfileField",
"id": "165",
"name": "Total Time",
"dataType": "number",
"internalName": "V_Total_Time",
"length": "4"
}, {
"type": "ProfileField",
"id": "166",
"name": "Total Visits",
"dataType": "number",
```

```
                "internalName": "V_Total_Visits",
                "length": "4"
                }, {
                "type": "ProfileField",
                "id": "257",
                "name": "Zip Code (from IP)",
                "dataType": "text",
                "internalName": "V_ZipCodeFromIP",
                "length": "50"
                }],
                "page": 1,
                "pageSize": 1000,
                "total": 26
        }
```


Platform notices

- We are modifying the Eloqua application API endpoints used to retrieve Campaigns, so that `memberCount` will now only appear if the request `depth` is set to `partial` or `complete`. Currently, `memberCount` is included if the request depth is set to `minimal`, `partial`, or `complete`. This change will be implemented in the Eloqua 484 release. Learn more

- **Update**: The Oracle Eloqua URL standardization that was to not renew the old certificates for secure.eloqua.com for POD 1 and www.02.secure.eloqua.com for POD 2 has been postponed indefinately.

  The certificates in question will be renewed so that traffic from the URLs with "legacy" naming conventions will continue to be redirected appropriately and there will be no disruption on your end. **As there is no longer an expiration date, any calls (API calls, redirects, etc) to the old URLs will continue to work**.

Learn more

~~Effective this coming Fall (2016), we will not be renewing the old certificates for secure.eloqua.com for POD 1 and www.02.secure.eloqua.com for POD 2, and as such clients must ensure they're pointing to the newer, updated, application URLs before they expire. Failure to do so before the certification expiration dates will result in any API calls to the old URLs failing. Certificate Expiration Dates:~~

- ~~POD 1 - Friday, November 18, 2016 for secure.eloqua.com~~
  - ~~https://secure.eloqua.com changes to https://secure.p01.eloqua.com.~~
- ~~POD 2 - Sunday, December 11, 2016 for www02.secure.eloqua.com~~
  - ~~https://www02.secure.eloqua.com changes to https://secure.p02.eloqua.com.~~ Learn more

Release 482

A list of developer-facing new features, significant recent changes, and platform notices for Oracle Eloqua release 482.

New features

Application API

- Added RESTful API support for Email sending to a single, or low volume of contacts, as well as the retrieval of associated deployment information. This was built to replace the same functionality, previously only supported via the SOAP API, which is to be deprecated.

  Example: Create and send an email deployment for an existing email asset to a single contact

  ```
  POST /assets/email/deployment
  ```

  Request body:

```
{
                                    "type": "EmailTestDeployment",
                                    "name": "REST Test 01",
                                    "contactId": "1",
                                    "email": {
                                    "type": "Email",
                                    "id": "100",
                                    "name": "test01"
                                    },
                                    "sendOptions": {
                                    "allowResend": "true",
                                    "allowSendToUnsubscribe": "false"
                                    }
                                }
```


Request response:


```
{
                                    "type": "EmailTestDeployment",
                                    "currentStatus": "normal",
                                    "id": "4",
                                    "depth": "complete",
                                    "name": "REST Test 01",
                                    "permissions": [
                                    "Retrieve",
                                    "SetSecurity",
                                    "Delete",
                                    "Update",
                                    "Activate"
```

```
                                    ],
                                    "email": {
                                    "type": "Email",
                                    "x_e10_previewUrl":
"https://p03.eloquapreview.com/Preview.aspx?siteId=238011564&userGuid=a24753
11-77bb-41e6-9db1-45c8c6402efa",
                                    "x_e10_previewExpiryDate": "1468620740",
                                    "x_e10_isTemplate": "false",
                                    "x_e10_createdAt": "1468620436",
                                    "x_e10_createdBy": "11",
                                    "currentStatus": "Draft",
                                    "id": "100",
                                    "createdAt": "1468620389",
                                    "createdBy": "11",
                                    "depth": "complete",
                                    "folderId": "42",
                                    "name": "Test_Email",
                                    "permissions": [
                                    "Retrieve",
                                    "SetSecurity",
                                    "Delete",
                                    "Update"
                                    ],
                                    "updatedAt": "1468620436",
                                    "updatedBy": "11",
                                    "archive": "false",
                                    "bounceBackEmail": "APIUserSandbox@s238011564.m.en25.com",
                                    "contentSections": [],
                                    "dynamicContents": [],
```

```
                                    "emailFooterId": "1",
                                    "emailGroupId": "4",
                                    "emailHeaderId": "1",
                                    "encodingId": "3",
                                    "fieldMerges": [],
                                    "forms": [],
                                    "htmlContent": {
                                    "type": "RawHtmlContent",
                                    "contentSource": "upload",
                                    "html": "<!DOCTYPE html> \r\n<html>\r\n  <head>  </head>\r\n
<p>Test Email</p>\r\n  </body> \r\n</html>"
                                    },
                                    "hyperlinks": [],
                                    "images": [],
                                    "isContentProtected": "false",
                                    "isPlainTextEditable": "false",
                                    "isPrivate": "False",
                                    "isTracked": "true",
                                    "layout": "{}",
                                    "plainText": "\r\nTest Email\r\n\r\n",
                                    "renderMode": "Flow",
                                    "replyToEmail": "API.User@test.oracle.com",
                                    "replyToName": "API User",
                                    "sendPlainTextOnly": "false",
                                    "senderEmail": "API.User@test.oracle.com",
                                    "senderName": "API User",
                                    "style": "{}",
                                    "subject": "Test"
                                    },
```

```
                "failedSendCount": "0",
                "successfulSendCount": "0",
                "contactId": "1",
                "sendOptions": {
                "type": "EmailSendOptions",
                "allowResend": "true",
                "allowSendToBounceback": "false",
                "allowSendToGroupUnsubscribe": "true",
                "allowSendToMasterExclude": "false",
                "allowSendToUnsubscribe": "false"
                }
        }
```


AppCloud Developer Framework

- Added an endpoint to retrieve an app's outbound logs. Leveraging this endpoint to view outbound logs enables app providers to retrieve logs for apps across different pods, while the logs page in Eloqua's web interface is limited only to displaying outbound requests for your instance and other clients on your pod. Learn more

  See an example

  Request body:

  ```
  GET /api/cloud/1.0/apps/20991738-0156-496d-nc41-9214219888d0/logs?limit=1
  ```

  Request response:

```
{
                                        "items": [{
                                        "requestId": 50,
                                        "appId": "20991738-0156-496d-nc41-9214219888d0",
                                        "requestDate": "2016-07-05T11:50:16.8230000",
                                        "requestMethod": "POST",
                                        "serviceInstanceId": "39baeb27-1234-5678-a073-87d17c4e1963",
                                        "serviceId": 75,
                                        "requestUrl": "https://localhost:12345/api/service/action/notify/3
1234-5678-a073-87d17c4e1963/4/Campaign?entityType=Contacts",
                                        "responseCode": 204,
                                        "details": {
                                        "requestHeader": "{\"Content-
Type\":\"application/json\",\"Accept\":\"application/json\",\"Accept-
Encoding\":\"gzip,
deflate\",\"Link\":\"<https://eloqua.example.com/api/bulk/2.0/syncs/2>;
rel=\\\"related\\\",
<https://eloqua.example.com/api/bulk/2.0/contacts/exports/2/data?offset=0&limi
t=1000>; rel=\\\"self\\\"\"}",
                                        "requestBody": "
{\"totalResults\":1,\"limit\":1000,\"offset\":0,\"count\":1,\"hasMore\":false,\"item
s\":[{\"Id\":\"1\",\"EmailAddress\":\"api.user@test.oracle.com\"}]}",
                                        "responseHeader": "{\"Pragma\":\"no-cache\",\"X-SourceFiles\":\
8?B?RDpcUm9ndWVcc3JjXE5lYnVsYS5Sb2d1ZVxhcGlcc2VydmljZVxhY3Rpb25cbm90
aWZ5XDM5YmFlYjI3LTUzMDAtNDU4OS1hMDczLTg3ZDE3YzRlMTk2M1w0XENhbXB
haWdu?=\",\"Cache-Control\":\"no-cache\",\"Date\":\"Tue, 05 Jul 2016 15:50:20
GMT\",\"Expires\":\"-1\",\"Server\":\"Microsoft-IIS/10.0\",\"X-AspNet-
Version\":\"4.0.30319\",\"X-Powered-By\":\"ASP.NET\"}",
                                        "responseBody": ""
```

```
                    }
                    },
                    }],
                    "totalResults": 10,
                    "limit": 1,
                    "offset": 0,
                    "count": 1,
                    "hasMore": true
                }
```

Authentication

- Modified the behaviour of OAuth so that each time a new access token is requested, a new refresh token will be returned. Learn more

Documentation

- We've launched our new API reference documentation at a new location in a new format.

  Our API reference documentation can now be found at:

  http://docs.oracle.com/cloud/latest/marketingcs_gs/OMCAC/

  See the New API Documentation Announcement to learn more.

Recent changes

Bulk API

- Including the read-only field `updatedAt` was causing imports to fail in two scenarios:
  - In a custom object import definition and data upload.
  - In a custom object import definition when `updatedAt` was left out of the data upload.

  Now, when including `updatedAt` in a custom object import definition and data upload, the import succeeds with the read-only field and data uploaded to the field being ignored. Also,

including `updatedAt` in a custom object import definition and leaving the field out of the data upload, will cause the import to return a warning, but still import the data successfully.

- Setting the import definition request parameter `isUpdatingMultipleMatchedRecords` to true resulted in newly created records appearing under created and updated in the logs.

Platform notices

- EloquaService API will remain supported until April 1st, 2017, with basic availability and security fixes only. Learn more
- ExternalActionService (aka Cloud Connector API) will continue to be supported indefinitely.