



Guide

Developers of Welect GmbH

© 2022 <https://www.de.welect.de/> Welect GmbH | Duisburger Straße 19 | 40477 Düsseldorf

Table of contents

1. Welect Integration - API for publisher	3
1.1 Welect Token	3
1.2 Best practices for integrating Welect	3
2. Multiwall (Adblockwall, Accesswall)	4
2.1 Best practices for integrating Welect	4
2.2 Rules	4
3. SDK	6
3.1 Integration	6
3.2 Preflight Checks	6
3.3 Run Welect's Ad Chooser	7
3.4 Post Ad Checks	7
3.5 API reference	7
4. Link	10
4.1 Exit	10
5. Integration API	12
5.1 Checking the availability of the Placement	12
5.2 Running the placement	13
5.3 Validating the token	13
6. Data API	14
6.1 General	14
6.2 Authentication	14
6.3 Data Requests	14



1. Welect Integration - API for publisher

This guide is for those who want to integrate Welect.

1.1 Welect Token

Welect has two different kinds of token:

1. Transaction token: Can be used to check whether a ticket was successfully finished
2. Temporary token: Is valid for a certain duration

1.2 Best practices for integrating Welect



To avoid common pitfalls here are some useful tricks and hints:

1. Avoid the Google Tag Manager - favour a script HTML tag and address Welect directly by the provided URL.
2. The script tag that contains your personal bundle URL should look like this:

```
<script src="https://static.wlct-one.de/p/bundles/<some-uuid>.js#wbss"></script>
```
3. Place the script tag at the end of the HTML body
4. Assure that the Welect script is loaded after the CMP script or execute Welect Script with a safety delay
5. Welect scripts do not depend on user consent - the loading and execution of Welect scripts should not depend on the result of consent manager providers.



2. Multiwall (Adblockwall, Accesswall)

A multiwall is a JavaScript bundle that evaluates various rules when loaded. It can be integrated directly into the HTML or added on demand via JavaScript. The multiwall takes care of:

- Ad availability check in the preflight phase
- Creation of dialogs
- Handling of tokens after a successful view

Example of a link for a multiwall bundle:

```
https://static.wlct-one.de/p/bundles/<some-uuid>.js#wbss
```

Your Welect manager should provide you with your own personal URL to your pre-configured mutliwall bundle.

2.1 Best practices for integrating Welect

To avoid common pitfalls here are some useful tricks and hints:

1. Avoid the Google Tag Manager - favour a script HTML tag and address Welect directly by the provided URL.
2. The script tag that contains your personal bundle URL should look like this:


```
<script src="https://static.wlct-one.de/p/bundles/<some-uuid>.js#wbss"></script>
```
3. Place the script tag at the end of the HTML body
4. Assure that the Welect script is loaded after the CMP script or execute Welect Script with a safety delay
5. Welect scripts do not depend on user consent - the loading and execution of Welect scripts should not depend on the result of consent manager providers.

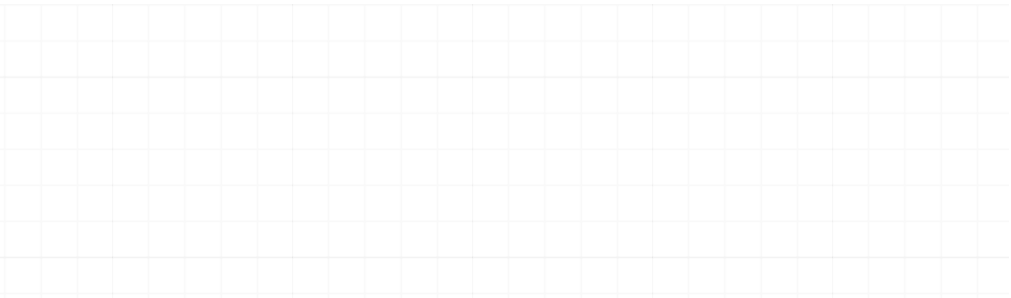
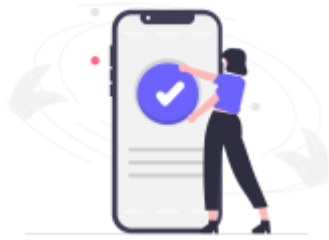
2.2 Rules

Rules can be separated into two categories. `Abort` rules prevent the ad chooser dialog and `Pick` rules do the opposite.

The following rules are predefined:

- Abort if valid token
- Abort if active cmp
- Pick if detection
- Pick if extended detection
- Pick if no consent
- Pick if negative consent
- Pick unconditionally - always trigger

Additional custom rules can be defined.





3. SDK

The SDK is a tool that gives the integrating partner the possibility to hook in and steer the Welect flow. It is a flexible approach that could be used in any web context. This quick introduction gives an overview of the important stages.

3.1 Integration

The SDK is an optimized JavaScript bundle. A Welect manager will provide you a link to this bundle. SDK bundles are configured individually for each partner or placement. But no worries - in any case the SDK bundle is ready to use and does not need any configuration from the SDK consumer side.

Example:

```
<!-- Embedded into the HTML of the serving site -->
<script src="https://static.welect.de/p/bundles/<a-bundle-id>.js#wbss"></script>
```

or

```
// Dynamic loaded
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = 'https://static.welect.de/p/bundles/<a-bundle-id>.js#wbss';
document.head.appendChild(script);
```

3.2 Preflight Checks

There are two things that could be checked in advance:

1. Has the user already completed a session and is this session still valid? (*)
2. Is any advertisement available that could be shown to the user?

The first point can be answered by using the `checkToken` method of the Welect object that will be provided via the SDK. And the second point can be checked with the `checkAvailability` method. Both methods accept callbacks that will be called either in the success or failure case.



The availability check (`checkAvailability`) can be called many times but it must be called at least once before continuing the Welect process.

(*It is possible to define the duration in which the user does not need to watch an ad again. For example 15 minutes)

3.3 Run Welect's Ad Chooser

The ad chooser executes the Welect process for selecting and playing an advertisement. This process can be triggered via the `runSession` method. It will take care of everything that belongs to this process. An alternative way is to use the `startURL` method. It returns an URL as a string. By opening this URL in any way in a browser window (tab or iframe) it will trigger the same flow.

3.4 Post Ad Checks

After running Welect's Ad Chooser a token will be returned. The token could be available either as query parameter or as a cookie. The query parameter is used in combination with a redirect and will be appended to redirect URL. The name of the parameter can be defined by Welect. The below specified `checkToken` method can analyze the token for validity.

3.5 API reference

Methods of `welect|Welect` objects:

```
interface CheckAvailabilityConfig {
  onAvailable: () => void;
  onUnavailable: () => void;
}

interface RunSessionConfig {
  onSuccess: () => void;
  onAbort: () => void;
}

interface CheckTokenConfig {
  onValid: () => void;
  onInvalid: () => void;
}

interface Welect {
  /**
   * Checks if any ads are available.
   */
  checkAvailability?: (config: CheckAvailabilityConfig) => void;
}
```

```

/**
 * Initiates the Welect overlway with its ad chooser.
 */
runSession?: (config: RunSessionConfig) => void;

/**
 * Analyzes the current window if a complete session is present.
 * A user has completed a session when the ad has been viewed till the end.
 */
checkToken?: (config: CheckTokenConfig) => void;

/**
 * Returns an URL which represents the Welect overlay with its ad
 * chooser.
 */
startURL?: () => string;
}

```

(Hint: To create the config objects use plain old JavaScript objects.)

3.5.1 Example

```

// Just a dummy method for this example!
const logFn = (event) => {
  return () => { console.log(event) };
};

// Has the user already watched a Welect ad?
Welect.checkToken({
  // Yes
  onValid: logFn('watched ad completely'),
  // No
  onInvalid: () => {
    // Are there any ads to show?
    Welect.checkAvailability({
      // No
      onUnavailable: logFn('no ad available'),
      // Yes
      onAvailable: () => {
        // Play the Welect dialog
        Welect.runSession({
          // Was the process interrupted by something?
          onAbort: logFn('view was cancelled'),
          // Ad view was completely
          onSuccess: () => {
            // Let`s check the token again
            Welect.checkToken({
              // Everything is fine
              onValid: logFn('watched ad completely'),
              // Please contact us in this case
              onInvalid: logFn('something went wrong'),
            })
          },
        })
      },
    });
  },
});

```

```
},  
});
```



4. Link

The Welect link is already completely configured and is ready to be used. The link starts and contains the whole Welect process. It can be used in an `<a>` Tag or called via JavaScript.

Example:

```
https://www.welect.de/passage/for_product/<our-id>/pre_offer
```

4.1 Exit

The exit mechanism can be configured in different ways.

4.1.1 When `redirect` is enabled

The predefined query parameter must be appended with the target url. The name can be individually defined by the Welect team. In this example the name of the query parameter is `origin` but it could also have a different name if required.

Example:

```
https://www.welect.de/passage/for_product/<our-id>/pre_offer?origin=https://www.welect.de
```

4.1.2 When `per token message` is enabled

This method is only useful if you are opening the link in an iframe.

Another exit mechanism is to send a message as an event. Any potential receiver of this message must be registered as an event listener.

Example:

```
const potentialReceiver = ({data}) => console.log(data);

window.addEventListener('message', potentialReceiver);
```

And this is what the receiver would get

```
interface TokenMessage {
  payload: string;
  payloadClass: string;
  type: string;
}
```

Example:

```
{ payload: "AdbhU5hveNhQjIivQkl_-IeYJAtiySsRoA==", payloadClass: "leaseToken", type: "welect:payload" }
```

4.1.3 Getting Token

In case of the exit method `token message` it is represented by the `payload` property of the data. Another way would be to add it as a query parameter with a predefined name. The name can be set by the Welect team.

In the example below the predefined name is `wlct-token`.

```
https://www.welect.de/?wlct-token=809982dd-f04b-445a-80c0-18ecb4eb6e8a
```



5. Integration API

As an alternative to our prebuilt Javascript [Multiwall](#) and [SDK](#) integration options that work completely in the browser of the user, we also offer an HTTP based API. Instead of having a time-based token that is valid for a certain amount of time after an ad was watched successfully, we provide a transactional token that can be validated exactly once. This is mostly only useful for integrations where the verification if an ad was watched completely should be done in the backend.

5.1 Checking the availability of the Placement

Similar to our prebuilt Javascript integrations this call has to be made from the client as our answer depends on the device type, the given consent within the TCF 2.0, etc.

```
POST https://www.welect.de/api/v2/preflight/<your-placement-id>
```

Please note that the parameters shown in the example below are required in the request body. It is important to note that although a valid tcString is required, there is no requirement about the consent that is represented by the tcString. Welect is able to deliver ads even if there is no consent given for any vendor.

If the placement is available the response will include an entry_url. Opening this url in a browser tab will start the placement.

5.1.1 Examples:

Request body:

```
{
  "gdpr_consent": {
    "gdprApplies": true, //true if GDPR regulations apply to this traffic
    "tcString": "CQB_d48QB_d48LDAAEENCZCAAAAAAAAAAAAAAAAAAAAA" //If gdpr applies a valid tc string has to be present
  },
  "query_parameters": {
    "payload": "https://www.example.com" //The payload is the link the user will be redirected to after watching an ad
  }
}
```

Response body if the placement is available:

```
{
  "available": true
  "entry_url": "https://welect.de/passage/pre_offer/38d4d74f-a350-48b7-9bfe-680e7e32595c?payload=https%3A%2F%2Fwww.examp1"
}
```

Please note that the `entry_url` will be different for every request.

If the placement is not available the response will be

```
{ "available": false }
```

5.2 Running the placement

In case an ad was watched successfully we will append a token via the `weselect-token` parameter that you can use to check if the user actually watched the ad.

If our placement is started with this:

```
https://weselect.de/passage/pre_offer/38d4d74f-a350-48b7-9bfe-680e7e32595c?payload=https%3A%2F%2Fwww.example.com
```

the user will be redirected to:

```
https://www.example.com?weselect-token=<some-token>
```

5.3 Validating the token

Validating the token should be done from your backend as it can only be done once per token. It is required to set an HTTP Header `x-api-secret` with a secret value that is provided per placement by Welect. Request:

```
PUT https://www.weselect.de/api/v2/ticket/<token>/close
```

Will result in a json response:

```
{
  success: true/false
}
```



6. Data API

We provide a JSON API to enable programatically collecting the data that is also available in the customer.welect.de Dashboard Application. An account for the Dashboard is necessary to use this API.

6.1 General

Every request should contain the following Header:

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

6.2 Authentication

Authentication for all requests is done via a Bearer Token. A Token can be acquired by the following request:

```
POST https://api.customer.welect.de/api/v1/session
```

The request body should contain the email and password from your Dashboard account, for example:

```
{
  "email": "jane.doe@example.com"
  "password": "my-safe-password"
}
```

A response to this request will look like this:

```
{
  "user": {...user information...}
  "token": <your-token>
}
```

The token you receive should be used as a Bearer Token for all other requests, so the following Header has to be added to all following requests:

```
Authorization: Bearer <your-token>
```

6.3 Data Requests

Data is being provided for both publishers and advertisers.

6.3.1 Publisher - Placement Overview

This request will give you a list of all currently active placements registered by us.

```
GET https://api.customer.weselect.de/api/v1/placements?archived=false
```

A response will look like this:

```
[
  {
    "publicId": "<unique-placement-id>",
    "name": "<placement-name>",
    "description": "...",
    "category": "...",
    "isArchived": false,
    "publisher": {
      ...
    }
  },
  ...
]
```

The `publicId` of each placement can be used to gather specific data for each placement.

6.3.2 Publisher - Placement Data

This request fetches all available data for the specified placements.

```
POST https://api.customer.weselect.de/api/v1/events/placements
```

The following parameters have to be used in the json request body:

Parameter Description

from	Specifies the date (YYYY-MM-DD) from which the data should be fetched
to	Specifies the date (YYYY-MM-DD) until where the data should be fetched
ids	A list of placement publicIDs acquired in the Placement Overview request
truncateTo	Either "days" or "months", the data will be accumulated for each day/month in the range between the specified from and to date

An example request body looks like this:

```
{
  "from": "2023-05-12",
  "to": "2023-05-14",
  "ids": ["<unique-placement-id1>", "<unique-placement-id2>"],
  "truncateTo": "days"
}
```

An example response looks like this:

```
[
  {
    "events": [
      {
        "eventName": "placement_start",
        "timebuckets": [
          {
            "timestamp": "2023-05-12",
            "weight": 1234
          },
          {
            "timestamp": "2023-05-13",
            "weight": 1234
          },
          {
            "timestamp": "2023-05-14",
            "weight": 1234
          }
        ],
        "type": "integer"
      },
      ...all other placement events...
    ],
    "publicId": "<unique-placement-id1>",
    "publisherPublicId": "<unique-publisher-id>"
  },
  {
    "events": [
      {
        "eventName": "placement_start",
        "timebuckets": [
          {
            "timestamp": "2023-05-12",
            "weight": 4321
          },
          ...
        ],
        "type": "integer"
      },
      ...
    ],
    "publicId": "<unique-placement-id2>",
    "publisherPublicId": "<unique-publisher-id>"
  }
]
```

6.3.3 Advertiser - Campaign Overview

This request will give you a list of all currently active campaigns registered by us.

```
GET https://api.customer.weselect.de/api/v1/campaigns?archived=false
```

A response will look like this:

```
[
  {
    "publicId": "<unique-campaign-id>",
    "name": "<campaign-name>",
    "isArchived": false,
    "publisher": {
      ...
    }
  },
  ...
]
```

The `publicId` of each campaign can be used to gather specific data for each campaign.

6.3.4 Advertiser - Campaign Data

This request fetches all available data for the specified campaigns.

```
POST https://api.customer.weselect.de/api/v1/events/campaigns
```

The following parameters have to be used in the json request body:

Parameter Description

from	Specifies the date (YYYY-MM-DD) from which the data should be fetched
to	Specifies the date (YYYY-MM-DD) until where the data should be fetched
ids	A list of campaign publicIDs acquired in the Campaign Overview request
truncateTo	Either "days" or "months", the data will be accumulated for each day/month in the range between the specified from and to date

An example request body looks like this:

```
{
  "from": "2023-05-12",
  "to": "2023-05-14",
  "ids": ["<unique-campaign-id1>", "<unique-campaign-id2>"],
  "truncateTo": "days"
}
```

An example response looks like this:

```
[
  {
    "events": [
      {
        "eventName": "consumption_started",
        "timebuckets": [
          {
            "timestamp": "2023-05-12",
            "weight": 1234
          },
          {
            ...
          }
        ]
      }
    ]
  }
]
```

```

        "timestamp": "2023-05-13",
        "weight": 1234
      },
      {
        "timestamp": "2023-05-14",
        "weight": 1234
      }
    ],
    "type": "integer"
  },
  ...all other campaign events...
],
"publicId": "<unique-campaign-id>",
"advertisementPublicIds": ["<unique-advertisement-id1>", "<unique-advertisement-id2>"],
"advertiserPublicId": "<unique-advertiser-id>"
},
{
  "events": [
    {
      "eventName": "consumption_started",
      "timebuckets": [
        {
          "timestamp": "2023-05-12",
          "weight": 4321
        },
        ...
      ],
      "type": "integer"
    },
    ...
  ],
  "publicId": "<unique-campaign-id2>",
  "advertisementPublicIds": ["<unique-advertisement-id1>", "<unique-advertisement-id2>"],
  "advertiserPublicId": "<unique-advertiser-id>"
}
]

```

6.3.5 Advertiser - Advertisement Overview

This request will give you a list of all advertisements registered by us.

```
GET https://api.customer.weselect.de/api/v1/advertisements
```

A response will look like this:

```

[
  {
    "publicId": "<unique-advertisement-id>",
    "name": "<advertisement-name>",
    "campaign": {
      ...
    }
  },
  ...
]

```

The `publicId` of each advertisement can be used to gather specific data for each advertisement.

6.3.6 Advertiser - Advertisement Data

This request fetches all available data for the specified advertisements.

```
POST https://api.customer.weselect.de/api/v1/events/advertisements
```

The following parameters have to be used in the json request body:

Parameter Description

from	Specifies the date (YYYY-MM-DD) from which the data should be fetched
to	Specifies the date (YYYY-MM-DD) until where the data should be fetched
ids	A list of advertisement publicIDs acquired either from Campaign Data or Advertisement Overview requests
truncateTo	Either "days" or "months", the data will be accumulated for each day/month in the range between the specified from and to date

An example request body looks like this:

```
{
  "from": "2023-05-12",
  "to": "2023-05-14",
  "ids": ["<unique-advertisement-id1>", "<unique-advertisement-id2>"],
  "truncateTo": "days"
}
```

An example response looks like this:

```
[
  {
    "events": [
      {
        "eventName": "consumption_started",
        "timebuckets": [
          {
            "timestamp": "2023-05-12",
            "weight": 1234
          },
          {
            "timestamp": "2023-05-13",
            "weight": 1234
          },
          {
            "timestamp": "2023-05-14",
            "weight": 1234
          }
        ],
        "type": "integer"
      },
      ...all other advertisement events...
    ],
    "publicId": "<unique-advertisement-id1>",
    "campaignPublicId": "<unique-campaign-id>"
  }
]
```

```

},
{
  "events": [
    {
      "eventName": "consumption_started",
      "timebuckets": [
        {
          "timestamp": "2023-05-12",
          "weight": 4321
        },
        ...
      ],
      "type": "integer"
    },
    ...
  ],
  "publicId": "<unique-advertisement-id2>",
  "campaignPublicId": "<unique-campaign-id>"
}
]

```