

[Documentation](#)[Core Concepts](#)[Advanced Topics](#)**Technical Guides**[Login](#)[Open Graph](#)[FQL](#)[Mobile App Install Ads](#)[Sample Apps](#)[API Reference](#)[SDK Reference](#)**Getting Started**[Login with Javascript SDK](#)**General**[Login as an App](#)[Login as a Page](#)[Extending Access Tokens](#)[Debugging Access Tokens and Handling Errors](#)[Handling Revoked Permissions](#)**Using the signed_request Parameter****Client-side**[Reauthentication](#)[Logging Users Out of Your App](#)[Client-Side Login Without Using the Javascript SDK](#)**Server-side**[Login for Server-side Apps](#)[Re-authentication](#)[Logout with the server-side login flow](#)**Other platforms**[Login for Desktop Apps](#)[Login for Canvas Apps](#)[Login for Mobile Web Apps](#)[Login for Devices](#)

Using the signed_request Parameter

[Technical Guides](#) > [Login](#) > Using the signed_request Parameter

A **signed_request** parameter is used by Facebook to pass data to an app in a number of different scenarios. It contains the `user_id` and other basic information such as an age range, a locale and a country - useful details for implementing in-app demographic restrictions.

This guide will walk you through the following:

- [Prerequisites](#)
- [Sample Overview](#)
- [Step 1. Parse the signed_request](#)
- [Step 2. Verifying the signed_request](#)
- [Troubleshooting](#)
- [Additional Info](#)

Prerequisites

You will need to have a basic Facebook app that fits one of the following criteria so that it receives the signed_request parameter:

- **Canvas Apps:** `signed_request` is POSTed to an app when it is loaded inside a [Canvas page](#).
- **Page Apps:** `signed_request` is POSTed to an app when it is loaded inside a [Page Tab](#).
- **Apps with Deauthorized Callback URL set:** `signed_request` parameter will be POSTed to the Deauthorized Callback URL (which is added in the [App Dashboard settings](#)) whenever a user deauthorizes it in their [account settings](#).
- **Apps using Registration Plugin:** `signed_request` is passed to an app which is using the [Registration Plugin](#) whenever a user successfully registers.
- **Apps using Javascript SDK for Login:** When using the [Facebook Login with Javascript SDK Login flow](#), the `authResponse` object contains a `signed_request` property.

Sample Overview

At the end of this guide you will have a sample function that can be used in your app (if it fits the above criteria) to capture and parse the `signed_request` and to verify it securely.

Step 1. Parsing the signed_request

As [noted above](#) there are different methods under which the `signed_request` will be sent to your app, either as a request variable or as a property in a JSON object. In both cases, capturing the parameter is trivial. Here is an example in PHP:

```
if ($_REQUEST) {
    $signed_request = $_REQUEST['signed_request'];
} else {
    echo '$_REQUEST is empty';
}
```

And here's another example using the Javascript SDK:

```
FB.getLoginStatus(function(response) {
    if (response.status === 'connected') {
        var signedRequest = response.authResponse.signedRequest;
    }
});
```

Parsing the `signed_request` will yield a JSON object containing some data. The content and structure of this object differs depending on the scenario in which it was generated, however the process of parsing it remains the same. Here is an example of parsing in PHP:

```
function parse_signed_request($signed_request) {
    list($encoded_sig, $payload) = explode('.', $signed_request, 2);

    // decode the data
    $sig = base64_url_decode($encoded_sig);
```

```

return $data;
}

function base64_url_decode($input) {
    return base64_decode(strtr($input, '-_', '+/'));
}

```

This will produce a JSON object that looks something like this:

```

{
  "oauth_token": "...big long string...",
  "algorithm": "HMAC-SHA256",
  "expires": 1291840400,
  "issued_at": 1291836800,
  "registration": {
    "name": "Paul Tarjan",
    "email": "fb@paulisageek.com",
    "location": {
      "name": "San Francisco, California",
      "id": 114952118516947
    },
    "gender": "male",
    "birthday": "12/16/1985",
    "like": true,
    "phone": "555-123-4567",
    "anniversary": "2/14/1998",
    "captain": "K",
    "force": "jedi",
    "live": {
      "name": "Denver, Colorado",
      "id": 115590505119035
    }
  },
  "registration_metadata": {
    "fields": "[\n {'name':'name'},\n {'name':'email'},\n {'name':'location'},\n {'name':'gender'}\n]",
    "user_id": "218471"
  }
}

```

Step 2. Verifying the signed_request

When you receive a `signed_request`, it can be verified using your App Secret to ensure the request was sent by Facebook and not a third party. To do so, modify your `parse_signed_request` function from above, as follows:

```

function parse_signed_request($signed_request, $secret) {
    list($encoded_sig, $payload) = explode('.', $signed_request, 2);

    // decode the data
    $sig = base64_url_decode($encoded_sig);
    $data = json_decode(base64_url_decode($payload), true);

    if (strtoupper($data['algorithm']) !== 'HMAC-SHA256') {
        error_log('Unknown algorithm. Expected HMAC-SHA256');
        return null;
    }

    // Adding the verification of the signed_request below
    $expected_sig = hash_hmac('sha256', $payload, $secret, $raw = true);
    if ($sig !== $expected_sig) {
        error_log('Bad Signed JSON signature!');
        return null;
    }

    return $data;
}

```

This additional part of the function recreates the expected signature of the `signed_request` from the received contents and then compares it to the signature that was actually received.

invalidating its contents.

Troubleshooting

My JSON is missing a few characters and not decoding. What should I do?

You are probably using normal base64 decoding instead of base64url decoding. See [wikipedia](#).

Why this signature scheme?

Many different signature schemes have been proposed over time. OAuth 1.0 built a base string to be signed, which was problematic because it was hard to get the normalization rules right. Our fb_sig signing scheme made it really easy for applications to use the variables without checking the signature. And normalizing what you're signing has always been a problem with signatures. To deal with all these issues, we've built a signature scheme, based on the OAuth 2.0 proposal.

Why base64url encoding?

This way, the only 'weird' characters you have to deal with are -, _ and . which are all safe to be put in URL parameters, HTTP headers, or URL anchors. No more worrying about percent encoding, removing newlines or browser incompatibilities with anchor tag decoding.

Why is it called signed_request?

It's the part of the HTTP request that is signed. You can remember the signature comes first in both the name of the parameter and the parameter itself.

Why is there a '.' in the middle? Don't you know the length of the signature?

Different signature algorithms have different lengths, and you don't know which algorithm you are using until you decode the JSON.

Why don't you use hex encoding for the signature?

It is nice to be consistent. The signature and the payload are now encoded the exact same way.

Why is the signature first?

Doing a left split is usually easier than a right one. It also allows us to pursue other encodings, like hex for the signature and percent encoding for the payload.

Additional Info

- [Reference: Basic Info included in the signed_request](#)

Updated about 2 weeks ago