

Heartbeat API



- [□□□□](#)
- [Using the API](#)
 - [Sending Data to the Server](#)
 - [Receiving and Responding on the Server](#)

Heartbeat API ██████████ WordPress ██████████ API ██████████



Diagram illustrating the distribution of data across different components:

- Row 1: 15 boxes (10 labeled "tick", 5 labeled "15-120")
- Row 2: 10 boxes (5 labeled "jQuery", 5 labeled "admin-ajax")
- Row 3: 20 boxes (10 labeled "JSON", 10 unlabeled)
- Row 4: 10 boxes (10 labeled "jQuery", 0 unlabeled)

Heartbeat

1. JS heartbeat-send
2. PHP heartbeat_received
3. JS heartbeat-tick

Using the API

Using the heartbeat API requires two separate pieces of functionality: send and receive callbacks in JavaScript, and a server-side filter to process passed data in PHP.

Sending Data to the Server

When Heartbeat sends data to the server, you can include custom data. This can be any data you want to send to the server, or a simple true value to indicate you are expecting data.

```
jQuery( document ).on( 'heartbeat-send', function ( event, data ) {  
    // Add additional data to Heartbeat data.  
    data.myplugin_customfield = 'some_data';  
});
```

Receiving and Responding on the Server

On the server side, you can then detect this data, and add additional data to the response.

```
/**  
 * Receive Heartbeat data and respond.  
 *  
 * Processes data received via a Heartbeat request, and returns additional data to pass back  
to the front end.  
 *  
 * @param array $response Heartbeat response data to pass back to front end.  
 * @param array $data      Data received from the front end (unslashed).  
 *  
 * @return array  
 */  
function myplugin_receive_heartbeat( array $response, array $data ) {  
    // If we didn't receive our data, don't send any back.  
    if ( empty( $data['myplugin_customfield'] ) ) {  
        return $response;  
    }  
  
    // Calculate our data and pass it back. For this example, we'll hash it.  
    $received_data = $data['myplugin_customfield'];  
  
    $response['myplugin_customfield_hashed'] = sha1( $received_data );  
    return $response;  
}  
add_filter( 'heartbeat_received', 'myplugin_receive_heartbeat', 10, 2 );
```

Back on the frontend, you can then handle receiving this data back.

```
jQuery( document ).on( 'heartbeat-tick', function ( event, data ) {  
    // Check for our data, and use it.  
    if ( ! data.myplugin_customfield_hashed ) {
```

```
    return;
}

alert( 'The hash is ' + data.myplugin_customfield_hashed );
});
```

Not every feature will need all three of these steps. For example, if you don't need to send any data to the server, you can use just the latter two steps.

```
### #2
[] [] [] 2026-03-26 17:08:39 UTC
[] [] [] 2026-03-26 17:08:39 UTC
```