



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Deployment Guide

Website Instrumentation

12/20/2025

Website Instrumentation

Contents

- **1 Website Instrumentation**
 - 1.1 document.domain
 - 1.2 Co-browse Proxy
 - 1.3 Document Type
 - 1.4 Basic Instrumentation
 - 1.5 Built-in Chat Deprecation
 - 1.6 Enabling Console Logs
 - 1.7 Advanced Instrumentation

You must instrument your website to enable Genesys Co-browse. This means that every page accessible by your customers must include the Co-browse JavaScript code. This code must be on the following page types:

1. Pages referred through links on the website or reachable through the address bar.
2. Pages loaded in iframes, which are hosted inside the first type of page.

The Co-browse Javascript code can be added to the web pages of any website that uses mainstream web technologies such as PHP, Java, or .NET.

document.domain

By default, the Co-browse JavaScript explicitly sets the `document.domain` property on the customer side to allow synchronization of iframes loaded from another sub-domain. Co-browse always sets the `document.domain` property to the second-level domain. You can disable Co-browse's modification of `document.domain` using the [setDocumentDomain option](#) in the JavaScript Configuration API.

If the scripts on your website also explicitly set `document.domain` and the value is different than the value set by Co-browse, one of the attempts (either from your website or Co-browse) to set `document.domain` will be overridden. This could potentially be unsafe if your site allows third-party users to create their own sub-domains because it might enable those users to get scripting access to the site.

Co-browse Proxy

You can quickly get up and running with any website by using the proxy-based approach. This approach is an easy way to test Co-browse in a lab environment without modifying your existing site; however, it has significantly lower performance in terms of page loading on the "Customer" side. For details about setting up the proxy, see [Test with the Co-browse Proxy](#).

Document Type

Genesys Co-browse relies on modern browser features, if available, and does not work with older browsers such as Internet Explorer 8. To activate these features in browsers which support them, use the appropriate document type definition at the beginning of each page in your website.

For HTML5:

```
<!DOCTYPE html>
```

For Internet Explorer, a special meta tag can be added:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

For more information, see the following:

- [Specifying legacy document modes](#)
- [Interoperable HTML5 Quirks Mode in IE10](#)

Basic Instrumentation

Co-browse is shipped with two JavaScript applications that each enables different functionality on your website.

- `gcb.min.js` — The default Co-browse JavaScript application, which includes Chat and Co-browse functionality.
- `genesys.min.js` — The [Integrated JavaScript Application](#), which includes Chat, Co-browse, and Web Engagement functionality.

The rest of this section covers how to add the default Co-browse JavaScript application to your site using `gcb.min.js`. For information about how to add the Integrated JavaScript Application to your site, see the [integrated instrumentation snippet](#).

To enable Co-browse and Chat, you must add the default Co-browse instrumentation snippet before the closing `</head>` tag on your web pages:

```
<script>(function(d, s, id, o) {  
  var fs = d.getElementsByTagName(s)[0], e;  
  if (d.getElementById(id)) return;  
  e = d.createElement(s); e.id = id; e.src = o.src;  
  e.setAttribute('data-gcb-url', o.cbUrl);  
  fs.parentNode.insertBefore(e, fs);  
})(document, 'script', 'genesys-js', {  
  src: "<COBROWSE_SERVER_URL>/cobrowse/js/gcb.min.js",  
  cbUrl: "<COBROWSE_SERVER_URL>/cobrowse"  
});</script>
```

You can use the snippet above to enable Co-browse and Chat on your website, but make sure you update `<COBROWSE_SERVER_URL>`:

- To load the JavaScript from the Co-browse server, set the `src` parameter to the following:
`http(s) :<COBROWSE_HOST>[:<COBROWSE_PORT>]/cobrowse/js/gcb.min.js`
- To connect the JavaScript application to the Co-browse server, set the `cbUrl` parameter to the following:
`http(s) :<COBROWSE_HOST>[:<COBROWSE_PORT>]/cobrowse`

This is the URL of the Co-browse application. It may also be the URL of the load balancer or reverse proxy. To enable secure content synchronization between the Customer (end user) browser and the Co-browse Server, use an HTTPS-based URL and HTTPS port instead.

Starting with Co-browse 8.5.002+, the customer side always uses the URL in the JS instrumentation for `css-proxy` and `url-proxy`.

JavaScript does not contain private personal information and can be loaded using HTTP. There are pitfalls in both cases that must be taken into account.

Warning

If a website is HTTPS-based, the browser might block JavaScript loaded/executed using HTTP.

WebSockets

With WebSockets enabled, Genesys Co-browse uses either WebSockets (ws://) or secure WebSockets (wss://) depending on the protocol of your **cbUrl**. If the **cbUrl** uses http:// then Co-browse uses ws:// and Co-browse uses wss:// when the **cbUrl** uses https://. When using https:// in the **cbUrl**, you do not need any additional configuration to use secure WebSockets.

Example Instrumentation

Here's an example with values set for the src and cbUrl parameters:

```
<script>(function(d, s, id, o) {  
  var fs = d.getElementsByTagName(s)[0], e;  
  if (d.getElementById(id)) return;  
  e = d.createElement(s); e.id = id; e.src = o.src;  
  e.setAttribute('data-gcb-url', o.cbUrl);  
  fs.parentNode.insertBefore(e, fs);  
})(document, 'script', 'genesys-js', {  
  src: "http://192.168.67.39:9700/cobrowse/js/gcb.min.js",  
  cbUrl: "http://192.168.67.39:9700/cobrowse"  
});</script>
```

The basic instrumentation snippet in the examples above is also part of the default instrumentation for the proxies (GWM and ZAP) that are included in the Co-browse Server installation package. Note that in the proxies <COBROWSE_SERVER_URL> is set to localhost:8700.

Tip

For more information about how to test the Co-browse solution using the proxy, refer to the [Test with the Co-browse Proxy](#).

Built-in Chat Deprecation

As of Co-browse **8.5.100.11**, support for the built-in chat is deprecated and the built-in chat button is hidden by default. You should use **Genesys Widgets** instead of the built-in chat widget.

To enable the deprecated built-in chat button, set the following:

```
<script>  
var _genesys = {  
  buttons: {
```

```
    chat: true
  }
};
</script>
```

Enabling Console Logs

All logging for the Co-browse JavaScript apps is turned off by default, but it can be enabled on both the customer side and agent side.

Enabling Console Logs on the Customer Side

This is done via the **debug** configuration option. Add this script before your instrumentation:

```
<script>
var _genesys = {
    debug: true
};
</script>
<script>(function(d, s, id, o) {
```

Enabling console logs on the agent side

Add the debug=1 parameter to the URL. For example:
<http://cobrowse:8700/cobrowse/slave.html#sid=123456789&debug=1>.

Advanced Instrumentation

To customize instrumentation and configuration of Co-browse, see the Co-browse [JavaScript API](#).