Developer's Guide

Genesys Co-browse 8.5.0

1/4/2022

Table of Contents

Genesys Co-browse 8.5 Developer's Guide

Welcome to the *Genesys Co-browse 8.5 Developer's Guide*. This document introduces you to the architecture of Genesys Co-browse and provides information about how you can customize it for your website. See the summary of chapters below.

Integrate with Agent Applications

Co-browse Monitoring Overview

Co-browse Server performance counters (KPIs)

Co-browse Monitoring Alarms

Product Overview

What is Genesys Co-browse?

Overview

Genesys Co-browse provides the ability for an agent and the end customer to browse and navigate the same web page at the same time. In a Genesys Co-browse session, both the agent and the customer share the same instance of the screen, as opposed to a conventional screen sharing application, where one of the parties sees an image of the other party's browser instance.

Components

Genesys Co-Browse is composed of the following components:

- **Genesys Co-browse Server** is a server-side component that is responsible for orchestrating the co-browsing activities between the end consumer and the agent.
- **Genesys Co-browse Plug-in for Interaction Workspace** provides co-browsing functionality for Interaction Workspace users.
- **Genesys Co-browse Plug-in for Workspace Desktop Edition** provides co-browsing functionality for Workspace Desktop Edition users.
- **Genesys Co-browse Sample Reporting Templates** provides configuration files and reporting templates for getting real-time and historical statistic data.
- **Integrated JavaScript Application** is a JavaScript component that includes the Chat, Co-browse, and (optionally) Web Engagement Tracker JavaScript applications. You should add this component to the pages on your website where you want to enable co-browsing.

Features

Genesys Co-browse includes the following features:

- Active participation—both the agent and the customer have the ability to take control.
- Browsing always happens on the customer side.
- Administrators are able to restrict what the agent can do and see on the web page. The customer can easily identify which fields are masked from the agent. Administrators can easily specify which DOM elements (buttons, check boxes, and so on) the agent must not be able to control.
- Support for multiple browsers, cross-browser support, and same-browser support.

  Support for scenarios in which the agent and customer are using different browsers.

  Support for scenarios in which the agent and customer are using different versions of the same browser.

- The customer can co-browse without downloading or installing any plug-ins.

Browser Support

Genesys Co-browse supports the following browsers:

- Internet Explorer 9 and above (Windows)
- Firefox 17 and above (Windows, Linux, and Solaris)
- Safari 6 and above (Mac)
- Google Chrome (Windows)

Genesys recommends that you use Internet Explorer 10 or above on agent machines for improved synchronization speed due to Web Sockets support and better JavaScript engine performance.

Warning

We strongly advise against IE Conditional Comments.

Important

Interaction Workspace uses *only* Internet Explorer as the embedded browser for working with Co-browse sessions.

Related Components

Genesys Co-browse interacts with the following Genesys Products:

- Workspace Desktop Edition — The Genesys Co-browse Plug-in for Workspace Desktop Edition is required to interface Genesys Workspace Desktop Edition with Genesys Co-browse. This plug-in enables the agent to join and terminate a co-browsing session with a customer.
- Chat Server — An eServices component, Chat Server handles chat interactions between agents and web visitors.
- Genesys Web Engagement — If Genesys Web Engagement is installed, the chat widget can be integrated with Genesys Co-browse and used to initiate a co-browsing session.

For a full list of related components see the Related Components page.

Important

For supported operating systems and a list of other required/compatible non-Genesys components, see Genesys Co-browse in the *Genesys Supported Operating Environment Reference Guide*.

Genesys Co-browse 8.1.3+ and Previous Co-Browsing Solutions

The Genesys Co-browse 8.1.3+ solution should not be associated with the Web API Cobrowse Samples. These samples work with the old KANA-based Co-Browsing Server and do not work with this new Co-browse solution; however, you may use a chat interaction started from the Web API Chat Samples to initiate a new Co-browse session from an instrumented page with an agent.

Important

Genesys strongly recommends that you use the Co-browse chat widget to initiate a Co-browse session with an agent. This chat widget is designed to work with the new Co-browse solution in the most optimal way. Agents do not even need to paste the Co-browse session ID manually into their screen - the Co-browse page is opened automatically once the agent clicks the "Co-browsing" button during a live chat.

For more information about how to work with the Co-browse chat widget, see the following sections:

- Initiating a Co-browse session from an integrated chat
- Genesys Co-browse and Chat

For more information about how to work with external chats like the Web API Chat Samples, see:

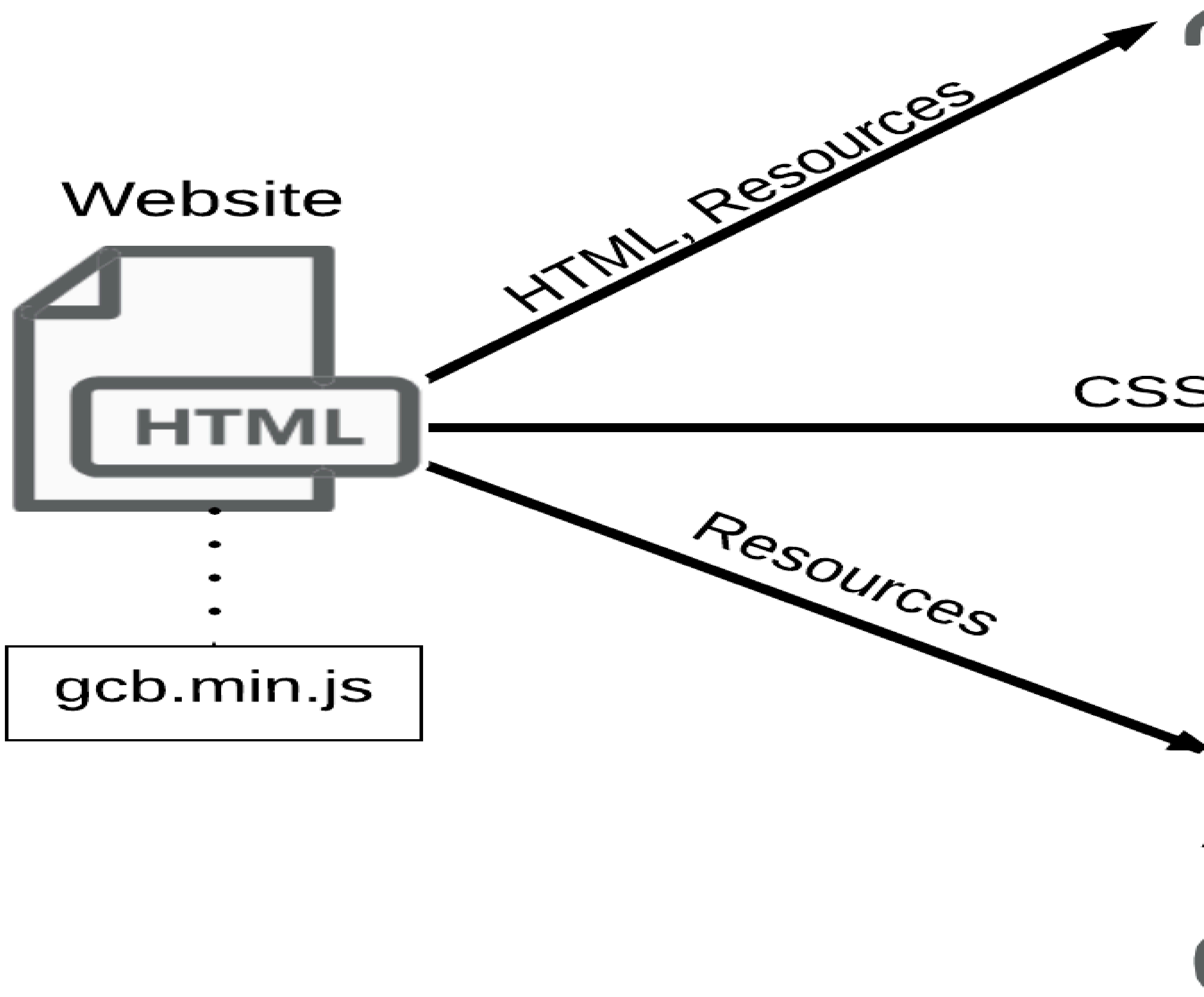
- Initiating a co-browse session from a voice call or external chat without integration
- External Chat without Integration

Restrictions and Known Limitations

See Co-browse Restrictions and Known Limitations.

Conceptual Model

Schematic Diagram

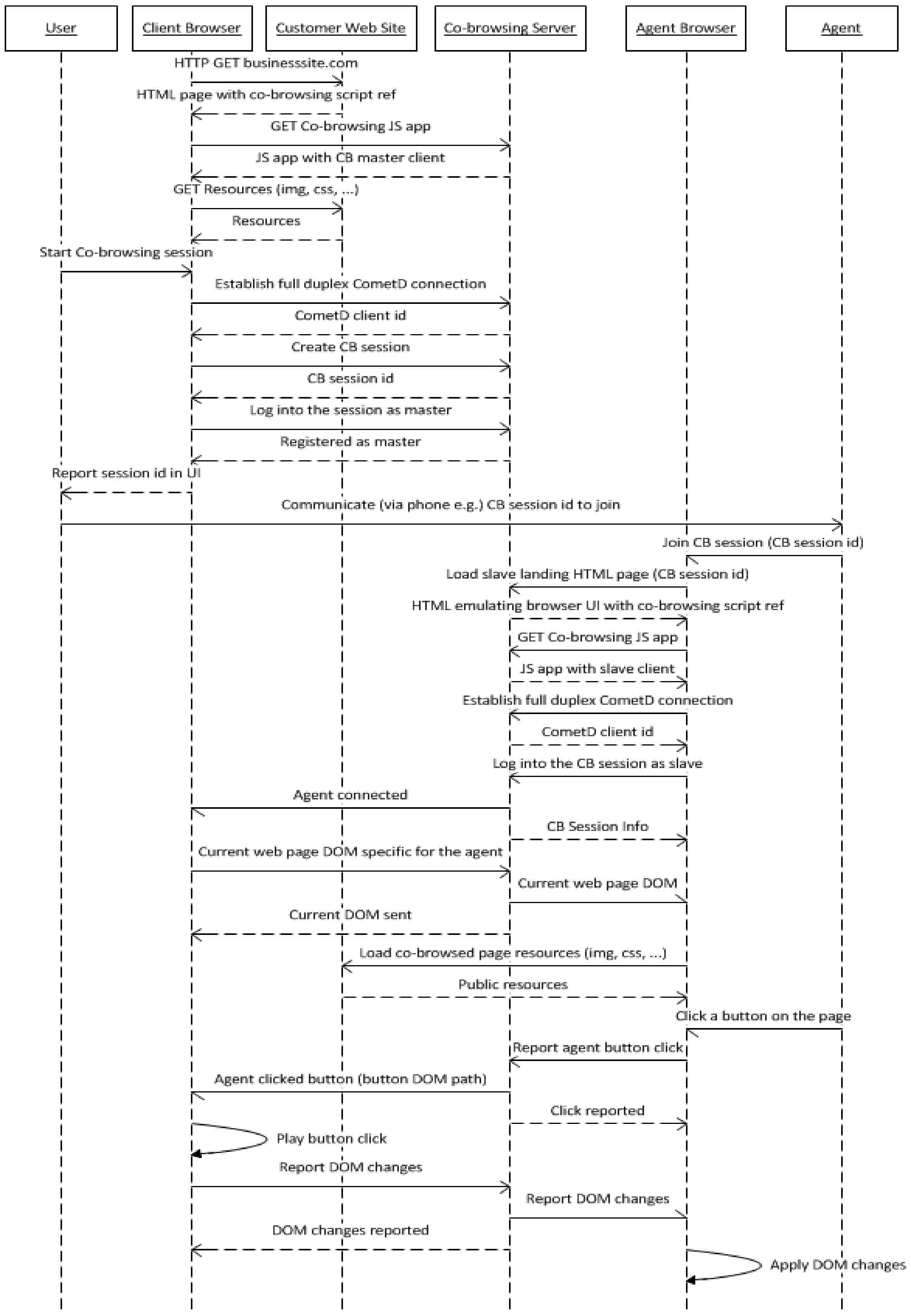

Website
HTML
gcb.min.js
HTML, Resources
CSS
Resources

After a user and an agent connect and a co-browsing session begins, several things start synchronizing.

One of the advantages of the approach shown in the schematic is that it allows for co-browsing to start at any time. For example, a user can initiate a co-browsing session while in the middle of filling out a long form. This approach also eliminates the need to worry about cookies and session timeouts on the agent side.

Sequence Diagram

The following diagram illustrates the detailed communication between the parties mentioned above.


User
Client Browser
Customer Web Site
Co-browsing Server
Agent Browser
Agent
HTTP GET businesssite.com
HTML page with co-browsing script ref
GET Co-browsing JS app
JS app with CB master client
GET Resources (img, css, ...)
Resources
Start Co-browsing session
Establish full duplex CometD connection
CometD client id
Create CB session
CB session id
Log into the session as master
Registered as master
Report session id in UI
Communicate (via phone e.g.) CB session id to join
Join CB session (CB session id)
Load slave landing HTML page (CB session id)
HTML emulating browser UI with co-browsing script ref
GET Co-browsing JS app
JS app with slave client
Establish full duplex CometD connection
CometD client id
Log into the CB session as slave
Agent connected
CB Session Info
Current web page DOM specific for the agent
Current web page DOM
Current DOM sent
Load co-browsed page resources (img, css, ...)
Public resources
Click a button on the page
Report agent button click
Agent clicked button (button DOM path)
Click reported
Play button click
Report DOM changes
Report DOM changes
DOM changes reported
Apply DOM changes

Sequence diagram

DOM Restrictions

Genesys Co-browse allows you to hide sensitive data from agents and restrict control of elements in a co-browse session by providing a map of the elements that should be restricted. For every element in this map, there should be a CSS selector which identifies it and the type of restriction applied.

You can implement two types of restrictions:

- Data masking
- DOM control

Important
Data masking and DOM controls apply to both Pointer Mode and Write Mode.

For details about DOM restrictions, see Configuring DOM Restrictions.

Data Masking

Private, critical, or sensitive customer-related data can be masked on the agent side with asterisks. This masked data does not leave the customer browser; it is not retrieved by Co-browse Server or the agent browser.

If agents try to change a masked input field, they see a notification that only the customer can access the input field.

Data masking can be applied to any visual HTML element. For input elements, the `value` attribute is masked; for all other elements, the text content is masked.

On the customers' side, if the data masked input field is in focus (for example, a customer selects an input field to enter a credit card number), customers see a notification that the information they type is not visible to agents.

Important
Data masking for all password inputs is enabled in the system and cannot be changed by Configuring DOM Restrictions.

DOM Control

DOM Control allows you to disable web page elements that agents should not be able to use. One example could be the `Sumbit` button for a web form.

> **Important**
> Buttons of type `submit` are disabled for agents by default.

Configuring DOM Restrictions

To implement DOM restrictions, you must create an XML file to store your configuration. In this XML file, you configure rules for data masking and DOM control. Sets of rules match to specific pages or groups of pages using regular expressions (regexp).

Creating an XML configuration file

Create an XML configuration with the following structure:

```
<?xml version="1.0" encoding="UTF-8" ?>
<domRestrictions>
    <restrictionsSet>
        <uriTemplate type="regexp" pattern="<URL matching regex>"/>
        <dataMasking>
            <element selector="..."/>
        </dataMasking>
        <domControl>
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

For a detailed example, see XML configuration file example.

After you create a DOM restrictions configuration file, you must link the XML configuration file to your Co-browse server.

XML structure description

The XML configuration file contains the following elements:

- `<domRestrictions>`—root tag containing any number of `restrictionsSet` tags.

  ```
  <domRestrictions>
      <restrictionsSet>
          ...
      </restrictionsSet>
      <restrictionsSet>
  ```

```
        ...
    </restrictionsSet>
</domRestrictions>
```

- `<restricitonsSet>`—defines restriction rules applied to a web page or group of pages matching the regular expression in the `pattern` attribute. Restriction sets are cumulative. More than one restriction set can apply to a single webpage.

```
<restrictionsSet>
    <uriTemplate type="regexp" pattern="..."/>
    <dataMasking>
        ...
    </dataMasking>
    <domControl>
        ...
    </domControl>
</restrictionsSet>
```

- `<uriTemplate>`—defines the set of web pages the restriction applies to using a URL matching regex pattern.

```
<uriTemplate type="regexp" pattern="<URL matching regex>"/>
```

`pattern` value:

- The `pattern` value is a regular expression (regex) that matches the URL of the target page. For more about regular expressions, see http://www.regular-expressions.info/.
- Some characters have special meaning in regular expression syntax. When using these characters, you must *escape* the characters using a backslash (\). URL characters you must escape: `.:()\/?*|+{}^$[]`.
- For example, the regex for the URI `http://www.genesys.com/about-genesys/contact-us` can be:

```
http:\/\/www\.genesys\.com\/about-genesys\/contact-us
```

or simply,

```
www\.genesys\.com\/about-genesys\/contact-us
```

Tip

You may use online tools like Regexper to validate your regular expressions.

`pattern` examples:

Regex	Description
`.*`	Matches any page
`login\.html`	Matches all pages that include `login.html` in the URL
`(login|registration)-page\.html`	Matches pages prefixed with `login` or `registration` such as `login-page.html` and `registration-page.html`.
`genesys\.com\/about-genesys\/contact-us`	Matches pages like `http://www.genesys.com/about-genesys/contact-us` and `https://genesys.com/about-genesys/`

Regex	Description
	contact-us.
(https\:\/\/)	Matches all HTTPS pages

- <dataMasking>—list of all web elements whose data should be masked.

```
<dataMasking>
    <element selector="..."/>
</dataMasking>
```

- <domControl>—list of all web elements that should be restricted from agent control.

```
<domControl>
    <element selector="..."/>
</domControl>
```

- <element>tag describing which element(s) to restrict.

```
<element selector="<jQuery specific selector>"/>
```

`element` examples:

Value	Description
`<element selector="#sendRequest"/>`	Element with id="sendRequest"
`<element selector="[name=login]"/>`	Element with name="login"
`<element selector="[type=submit]"//>`	Element with type="submit"
`<element selector=".SendButton"/>`	Elements with "SendButton" class
`<element selector="[href='/about-us/contacts']"/>`	Element with href="/about-us/contacts"
`<element selector="[href$='.org']"/>`	Elements with href attribute ending with ".org"
`<element selector=":button"/>`	All normal buttons
`<element selector="[type=submit]:not(#uniqueSubmitId)"/>`	All `submit` buttons without id = "uniqueSubmitId"
`<element selector=".Input:not(#InputId2)"/>`	All input fields in class `Input` and without id = "InputId2"

For more information about selecting elements from a webpage see Using browser tools to select an element.

XML structure summary

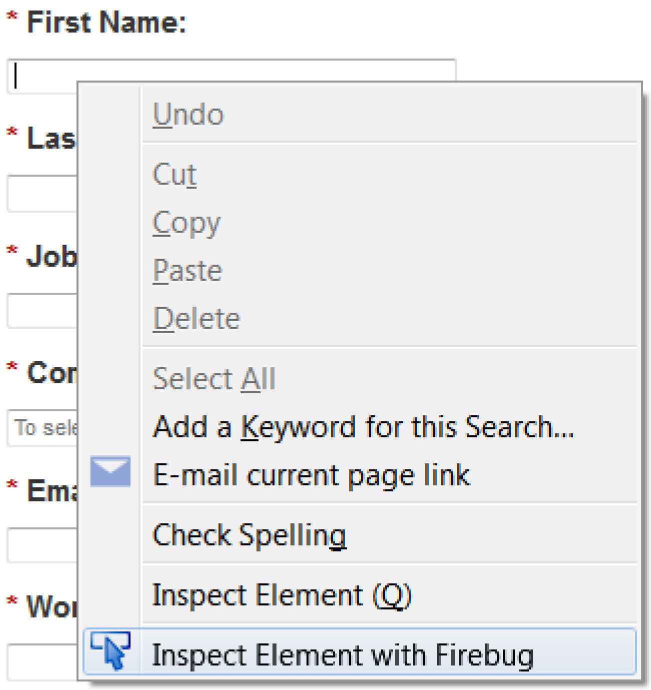
For each matched page or group of pages defined by a `pattern` in a `<uriTemplate>` tag, you can provide data masking and DOM control rules for specific page elements. Elements are represented by `<element>` tags. Each `<element>` contains a `selector` attribute with a jQuery selector describing the path to the element on the page.

Using browser tools to select an element

You can use browser tools to help you define an `element selector` in your XML configuration files. The example below uses the Firebug browser tool for Firefox.

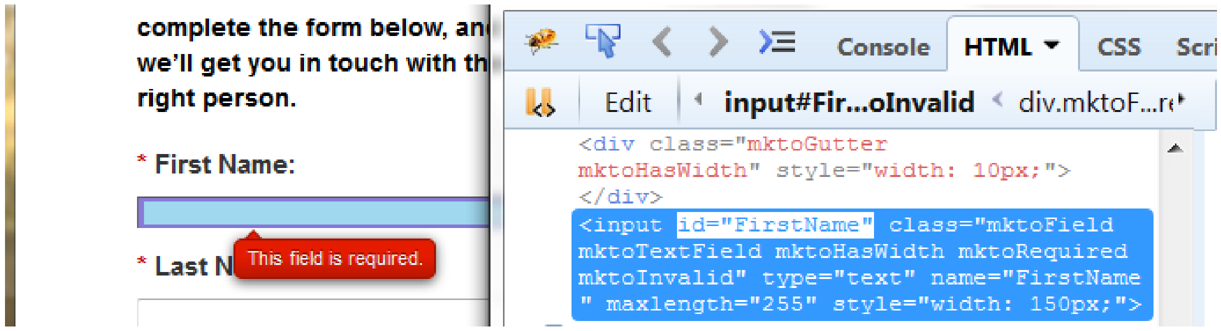
Using Firebug to find an element selector

1. Open the webpage containing the element you want to select. Right click on the element and click `Inspect Element with Firebug` to open the Firebug tool.



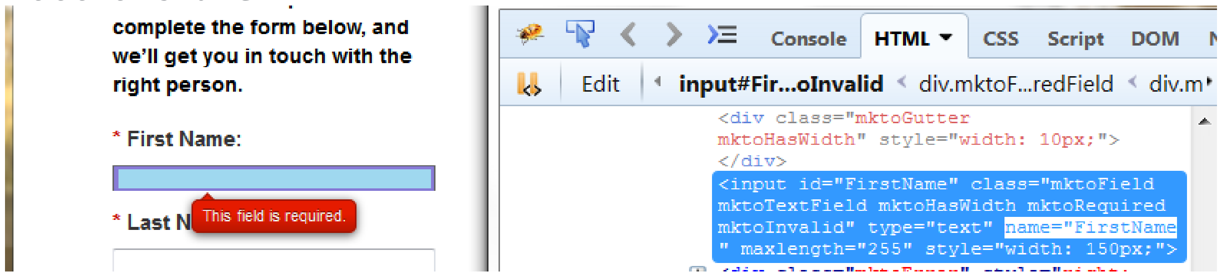
2. Firebug can help you identify the right selector for an element. For example, you could use one of the following as a selector:

   - The element's **id:**



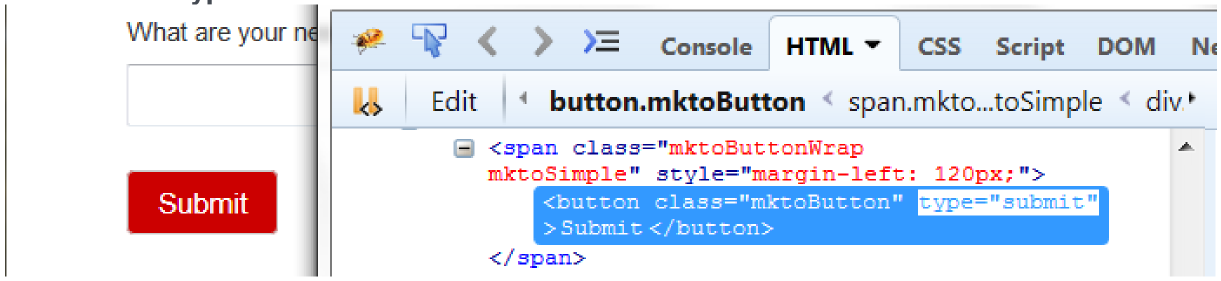
```
<div class="mktoGutter
mktoHasWidth" style="width: 10px;">
</div>
<input id="FirstName" class="mktoField
mktoTextField mktoHasWidth mktoRequired
mktoInvalid" type="text" name="FirstName
" maxlength="255" style="width: 150px;">
```

   - The element's **name:**



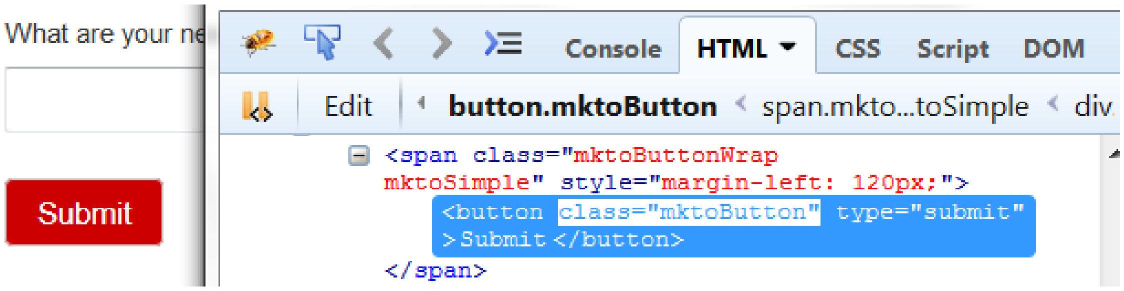
```
<div class="mktoGutter
mktoHasWidth" style="width: 10px;">
</div>
<input id="FirstName" class="mktoField
mktoTextField mktoHasWidth mktoRequired
mktoInvalid" type="text" name="FirstName
" maxlength="255" style="width: 150px;">
```

- The element's **type:**



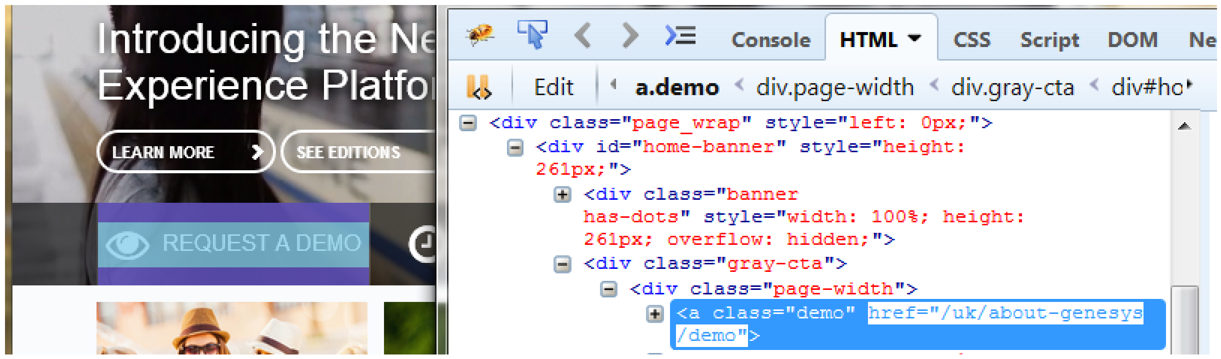
```
<span class="mktoButtonWrap
mktoSimple" style="margin-left: 120px;">
    <button class="mktoButton" type="submit"
    >Submit</button>
</span>
```

- The element's **class:**



```
<span class="mktoButtonWrap
mktoSimple" style="margin-left: 120px;">
    <button class="mktoButton" type="submit"
    >Submit</button>
</span>
```

- The element's **href:**



```
<div class="page_wrap" style="left: 0px;">
    <div id="home-banner" style="height:
    261px;">
        <div class="banner
        has-dots" style="width: 100%; height:
        261px; overflow: hidden;">
        <div class="gray-cta">
            <div class="page-width">
                <a class="demo" href="/uk/about-genesys
                /demo">
```

Tip

For a comprehensive list of JQuery selectors, see http://api.jquery.com/category/selectors/.

Default DOM Restrictions Configuration

The default configuration ensures DOM control for all submit buttons is restricted from agents.

Important

To preserve this default behavior, create your custom configuration by extending and not overwriting the default configuration.

```
<?xml version="1.0" encoding="UTF-8" ?>
<domRestrictions>
    <restrictionsSet>
        <uriTemplate type="regexp" pattern=".*"/>
        <domControl>
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

Important

Data masking for all password inputs is enabled in the system and cannot be changed using DOM restrictions configuration.

XML configuration file example

The example below provides a sample configuration with comments explaining the purpose of each element.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Use this configuration as a set of examples and common documentation -->
<!--
    domRestrictions contains any number of restrictionsSet elements.
    There must be only one domRestriction element.
-->
<domRestrictions>
    <!--
        Each set defines restrictions rules matched by URL.

        Starting with including restrictionSet from default configuration,
        so that agents can never click submit buttons on behalf of customers.
    -->
    <restrictionsSet>
        <!-- Pattern ".*" matches any string and therefore any URL -->
        <uriTemplate type="regexp" pattern=".*"/>
        <domControl>
            <!-- All submit buttons (elements with type = "submit" ) will be restricted -->
            <element selector="[type=submit]"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
    <!--
         All domControl and dataMasking rules in this restrictionSet will apply
        to pages that have "page.html" in their URL
    -->
    <restrictionsSet>
        <uriTemplate type="regexp" pattern="page\.html"/>
        <domControl>
            <element selector=":button"/> <!-- All normal buttons -->
            <element selector="#mySubmitButton"/> <!-- Concrete element with id =
"mySubmitButton" -->
            <element selector=".MySubmitButton"/> <!-- All elements that have class
"MySubmitButton" -->
            <element selector="[href='/checkout']"/> <!-- All links that lead to /checkout
page -->
        </domControl>
        <dataMasking>
            <element selector="[name=login]"/> <!-- All elements with name="login" will be
masked -->
            <element selector=".LicenseCode"/> <!-- All elements with class "LicenseCode"
will be masked -->
        </dataMasking>
    </restrictionsSet>
    <restrictionsSet>
        <!-- Range of pages pattern. -->
        <uriTemplate type="regexp" pattern="genesys\.com\/page[1-9]\.html"/>
        <domControl>
            <!--  Element with id = "uniqueSubmitId" will be excluded from restriction -->
            <element selector="[type=submit]:not(#uniqueSubmitId)"/>
        </domControl>
        <dataMasking/>
    </restrictionsSet>
</domRestrictions>
```

Linking the XML configuration file to your Co-browse Server

After you have created your configuration file, set the value of the domRestrictionsURL configuration option from the session section to point to your file. You can configure this value as:

- a URL reachable by Co-browser Server. For example, `domRestrictionsURL=http://cobrowse.com/static/dom_restrictions.xml`.
- a path to the file pre-fixed with `file`. For example, `domRestrictionsURL=file:C:\restrictions.xml`.

Integration with Agent Applications

Overview

Co-browse functionality is integrated with an agent application in two steps:

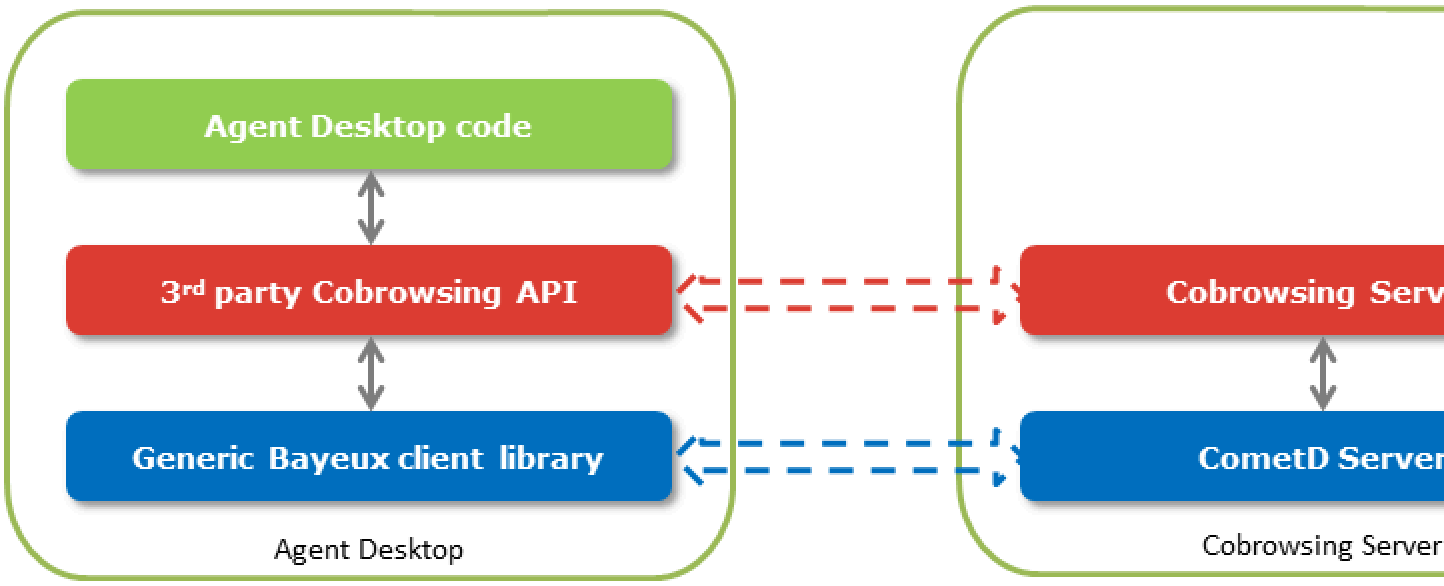
1. The Co-browse agent web UI is integrated with an agent application UI in one of two ways:
   - A browser (embedded) for desktop applications such as Interaction Workspace.
   - An iframe for web applications.
2. The agent application independently connects to the Co-browse Server in order to:
   - Control the co-browse session.
   - Integrate the co-browse session with the primary interaction (chat or voice).
   - Attach Co-browse data to the primary interaction.

Connection to Co-browse Server

Co-browse Server allows clients to inter-operate within a Co-browse session, mainly using a CometD-based API. The connected clients have one of the following roles in the session:

1. Customer — JavaScript client that shares the source web page.
2. Agent — JavaScript client that creates a remote view of the shared page.
3. Controller — Agent application that controls the Co-browse session.

Controller to Co-browse Server Connection



Important
Language-specific libraries for use inside agent applications to integrate with Co-browse Server (third-party Co-browsing API in the graphic above) have not been developed yet.

The agent application establishes a connection to the Co-browse Server through the third-party Co-browse API library (it will be built on top of a CometD .NET client and a Java client, for .NET and Java respectively) before loading the Co-browse agent UI. It supplies the Co-browse session access token, which is required to connect to an already created session. As a result, the agent application is connected to the session as the Controller.

Integration with Agent Desktop and Web-Based Applications

Integration with Agent Desktop Applications

In order to integrate an agent desktop application with Co-browse you must develop a plug-in that:

1. Serves the Co-browse agent page in a browser (Internet Explorer 9+/Firefox/Chrome).
2. Integrates with Co-browse Server (Controller connection).

To avoid race conditions when receiving session-activated notifications, the Co-browse plug-in should connect as Controller before opening an agent page in a browser. In future releases, the session join response may be extended to include session information (for example, session activation time).

Co-browse Web Agent Page

The agent page is the agent side of the co-browse functionality. The web agent page is opened in a browser with a dynamically built URL in the following format:

```
<schema>://<host>:<port>/cobrowse/slave.html#nosidinput=1&sid=<sessionToken>
```

`<schema>` — `http` or `https`

`<host>` — Co-browse service host (from Configuration Server)

`<port>` — Co-browse service port (from Configuration Server)

`<sessionToken>` — Co-browse session token. This token is transferred from user to agent by any communication channel.

`nosidinput=1` — Specify this parameter if the plug-in UI provides its own session ID input field.

> **Important**
> For improved security, `slave.html` parameters are passed in the fragment identifier (also known as hash) of the URL. Formerly, `slave.html` parameters, including session ID, were passes as query parameters. The query parameter method is still supported for backwards compatibility but is subject to removal in future releases.

Co-browse Server Integration

Server integration can be split into the following functions:

1. Chat integration — Intercept the "start co-browse" chat message (optional).
2. Join a co-browse session (mandatory).
3. Close a co-browse session (mandatory).

> **Important**
> Co-browse Server provides CometD and REST collaboration interfaces. The agent desktop application can use both. In order for it to work correctly, the application must

be subscribed to at least the JOIN, ACTIVATED, and DEACTIVATED CometD channels.

Chat Integration

This is optional functionality. It provides automatic co-browse session start by a specially formatted chat message. For this functionality, the agent desktop application must be able to:

1. "Listen" to the chat interaction (chat messages).
2. Check if a message matches a regular expression: `^\{start:\d{9}\}$`
3. If the message matches, parse the session token from the message (nine digits after "start:").
4. Join a co-browse session.

Join a Co-browse Session

A co-browse session can only be joined by having a session token (see Co-browse Web Agent Page) and completing the following steps:

1. Issue a request to the REST GET session method.
2. Use the reply to *stick* to the server that "owns" the session (see Stickiness). There are 2 ways of doing this:
    - Pass the `gcbSessionServer` cookie with the server name (`sessionServerName` in response) in all further HTTP requests (including CometD).
    - If the cookies are problematic, use the server URL (`sessionServerUrl` in response) for all further HTTP requests (including CometD).
      Note: For this to work you have to have the **serverUrl** option set for all Co-browse nodes in a cluster.
3. Establish a CometD connection.
4. Join a co-browse session as Controller by sending a message on the JOIN channel.
5. Wait for the async server response (notification) on the JOIN channel.
6. Attach the following to the current interaction:
    - `CoBrowseAssociated` — The co-browse session flag. This marks the interaction with an active/inactive co-browse session - "Yes" value.
    - `CoBrowseSessionsQuantity` — The number of co-browse sessions for the current interaction.
    - `CoBrowseSessionId` — The co-browse session history identifier received from the JOIN channel (`sessionHistoryId`).
7. Wait for the async server notification on the ACTIVATED channel.
8. Attach the following to the current interaction:
    - `CoBrowseStartTime` — A string with the co-browse session start time in UTC (session start time as a timestamp received from the ACTIVATED channel).

Close a Co-browse Session

A co-browse session may be closed due to the following reasons (click "[Show details]" for information about how to handle each scenario):

- The user or agent exits the co-browse session (via the Co-browse web UI) without closing the primary interaction. **[Show details]**

  In this scenario, complete the following steps:

  1. Wait for the async server notification on the DEACTIVATED channel.
  2. Attach the following to the current interaction:
     - `CoBrowseDuration` — A summary, in seconds, of the duration of all co-browse sessions (in case there were multiple co-browse session conducted within the same primary interaction).
     - `CoBrowseEndTime` — The current co-browse session end timestamp, as a string.
     - `CoBrowseAssociated` — The co-browse session flag. This marks the interaction with an active/ inactive co-browse session - "No" value.

- The primary interaction is transferred (Co-Browse Server currently does not support co-browse session transfer). **[Show details]**

  In this scenario, complete the following steps:

  1. Intercept the interaction transfer event.
  2. Send a synchronous (via REST) or asynchronous (via CometD) STOP command.
  3. Attach data to the current interaction based on data received in response to the stop command (see "The user or agent exits the co-browse session" above for details).
  4. Handle the deactivated notification to clean up resources (disconnect the CometD client, for example) in a unified way.

- The inactivity timeout has expired against the primary interaction (see Interaction Server's option `settings/handling-timeout`). **[Show details]**

  When the inactivity timeout expires against the primary interaction, then the interaction is taken away from the agent and placed back into the queue and routed once more.

  The Co-browse plug-in reaction is the same as the primary interaction transferred case (since the inactivity timeout case can be thought of as a transfer initiated by the system).

- The primary interaction is closed. **[Show details]**

  In this scenario, complete the following steps:

  1. Intercept the interaction closing event.
  2. Perform steps 2-4 in the "The primary interaction is transferred" section above.

> **Important**
>
> To attach data to the interaction, the agent application must attach the data **before** the interaction is actually closed.

Integration with Agent Web-based Applications

The web agent Co-browse application can be easily integrated with any web application, such as a web-based agent desktop. To do this, simply add an iframe with the web agent app into the web application:

```
<!doctype html>
<head>...</head>
<body>
...
<iframe src="http://<CB_SERVER>/cobrowse/slave.html"></iframe>
...
</body>
```

It is important to keep `slave.html` in the URI, even if you proxy the Co-browse Server, because the Co-browse scripts rely on it. For example, the URI http://my-site.com/cobrowseAgent/ **does not** work, even though it actually points to `slave.html`.

You can enable the web browser console logs in the standard way by adding the debug=1 URL parameter.

```
<iframe src="http://<CB_SERVER>/cobrowse/slave.html#debug=1"></iframe>
```

To have the web agent immediately join a session, create an iframe with a predefined `sid` URL parameter:

```
<iframe src="http://<CB_SERVER>/cobrowse/slave.html#sid=123456789"></iframe>
```

You should use the maximum possible size for the iframe because the Co-browse area adjusts to the end customer's browser window and can be big if the user has a large monitor, for example. If the Co-browse area becomes larger than the containing iframe, an agent will see scrollbars, which may not be very convenient.

Important

For improved security, `slave.html` parameters are passed in the fragment identifier (also known as hash) of the URL. Formerly, `slave.html` parameters, including session ID, were passes as query parameters. The query parameter method is still supported for backwards compatibility but is subject to removal in future releases.

Chat

Genesys Co-browse supports three levels of integration with chat functionality:

- Co-browse has its own built-in chat widget that is triggered with the "Live Chat" button. You can customize this built-in chat widget to suit your needs or you can implement your own chat widget using the Chat Service API.
- Co-browse can be used with any external chat without integration. In this case, the user will have to manually transfer the Co-browse session ID to the agent. See External Chat Without Integration.
- Co-browse can be integrated with an external JavaScript-based chat to use the Co-browse built-in "Live Chat" button to start a chat and to automatically transfer the Co-browse session ID to the agent. You will need to implement an External Media Adapter for your chat widget and pass your external media adapter object to the `primaryMedia` Co-browse configuration option.

External Chat Without Integration

Co-browse and non-integrated chat

If your website already has chat, you can use it with Genesys Co-browse without any other integration effort. For example, you may use the Web API Chat Samples to start a chat.

To allow users to initiate a Co-browse session from your website, you must add the Co-browse JavaScript snippet to your web pages. For information about how to enable it, see Website Instrumentation. By default, the instrumentation has a "Live Chat" button used for embedded initiation that is not needed if you already have chat enabled on your website. In this case, you should disable the built-in Chat. See Disabling Chat or Co-browse for details.

If your web page is correctly instrumented, the user will see a "Co-browsing" button after loading. Now, when the user clicks the "Co-browsing" button, they will see a notification message. If the user clicks "Yes", the Co-browse session will begin and the user will be prompted to manually transfer the session ID to the agent using either chat or voice.

Customize the Built-in Chat Widget

The built-in chat widget provides the main chat functionality for Genesys Co-browse. It's a versatile widget that can be customized through the Chat Service JS API and the Chat Widget JS API.

Customization

There are three different customization types available for modifying the chat widget UI: Template-based, CSS-based, and JavaScript-based. Using these customization types, you can do any of the following:

- modify the structure of the widget
- add content
- add css classes
- modify the style (including the logo and buttons)
- use JavaScript APIs to modify widget behavior

For details about the customization types and how you can use them, see Customizing the User Interface, part of the Chat Widget JS API.

Localization

You can use the `startChat` and `restoreChat` methods of the Chat Widget JS API to enable localization for the chat widget. For details and step-by-step instructions, see Localization.

Customize Genesys Co-browse User Interface

The Genesys Co-browse user interface widgets that are visible on your website are based on HTML, CSS, and JavaScript. You can customize these widgets to suit the look and feel of your website.

There are three ways you can customize the UI (User Interface):

- Adding additional CSS to your website that overrides the default CSS for the widgets. See Customizing the CSS below.
- Using the JavaScript API.
- Using Localization.

See Customization Examples for several detailed examples of UI customization.

Customizing the CSS

The Co-Browse JavaScript automatically loads some CSS files that define how the widgets look. To find these files, you can do one of the following:

- Download the complete CSS file from the URL `http(s)://<COBROWSE_HOST>[:<COBROWSE_PORT>]/cobrowse/css/master-all.css`. This is a good starting point for medium to broad customizations such as changing the color scheme.
- Use Firebug, Chrome Developer Tools, or a similar browser tool to select CSS rules for particular elements.

All Co-browse CSS follows the same principles:

- Only classes are used.
- All Co-browse classes begin with the `.gcb-` prefix.

You can override the Co-browse CSS by adding your own custom rules anywhere in the <head> tag.

Tip

Although the Co-browse CSS is loaded dynamically, it is prepended to the beginning of the <head> tag, so your custom CSS will always have higher specificity.

Customization Examples

See Customization Examples for detailed customization examples such as how to customize the Co-browsing button and the toolbar position.

Localization

See Localization.

UI Customization Examples

This is a collection of UI customization examples. Although these examples do not cover every possible customization case, they should give you a sample of what can be achieved and how to begin.

Tip
Some examples use Firebug as a developer tool. Any other similar tool can be used instead such as Google Chrome Developer Tools, Safari Web Inspector, or Internet Explorer Developer Tools.

Tip
To customize the built-in chat widget, see Customizing the Chat Widget User Interface.

Example: Customizing Live Chat and Co-browsing Buttons

Customization type: **CSS Based**

Prerequisites

You must have basic knowledge of CSS and HTML.

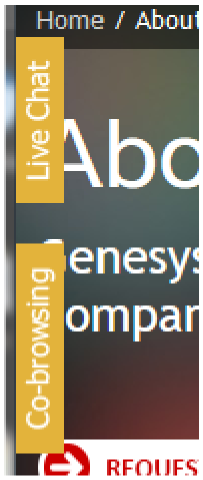
Tip
The Live Chat and Co-browsing buttons are images and cannot be modified using the Localization mechanism.

Tip
You can manage the visibility and moderately change the position of the default buttons using the JavaScript API.

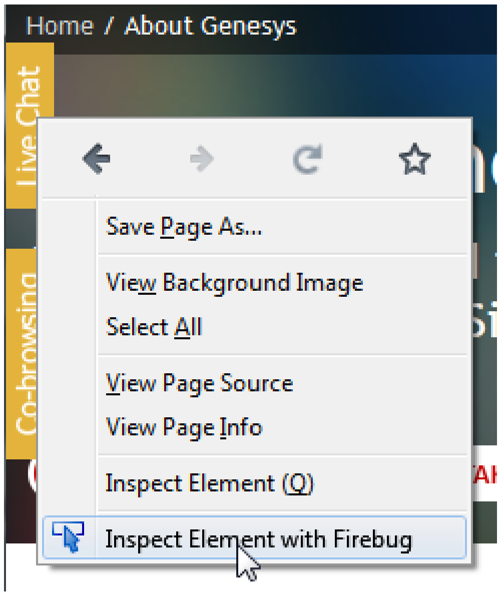
Start of Procedure

1. In Firefox, open the page instrumented with the Co-browse JavaScript application. Live Chat and Co-

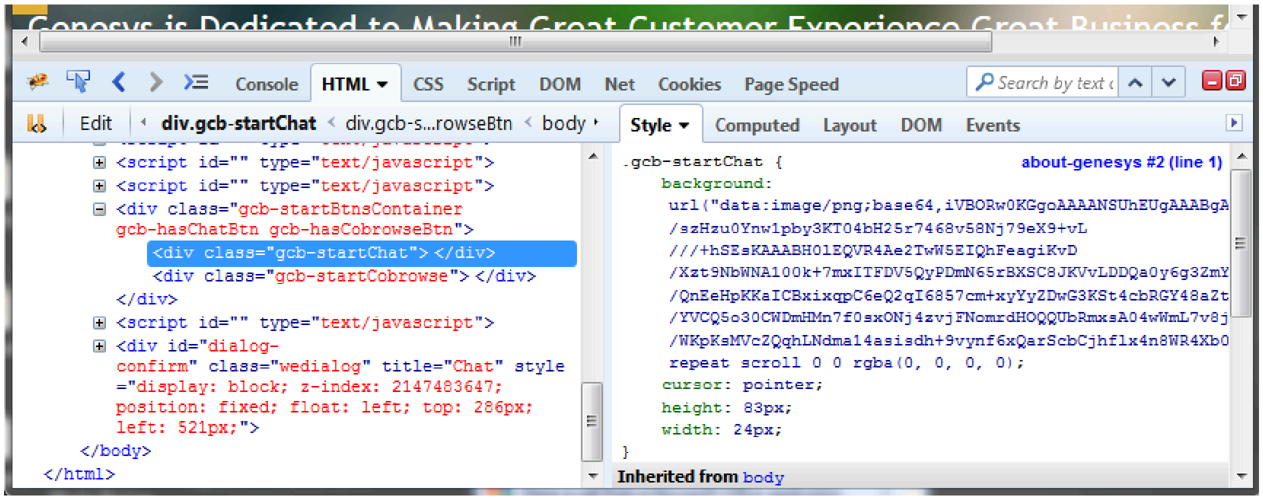
browsing buttons should appear:



2. Right click the `Live Chat` buttons and click `Inspect Element with Firebug` to open the Firebug tool.



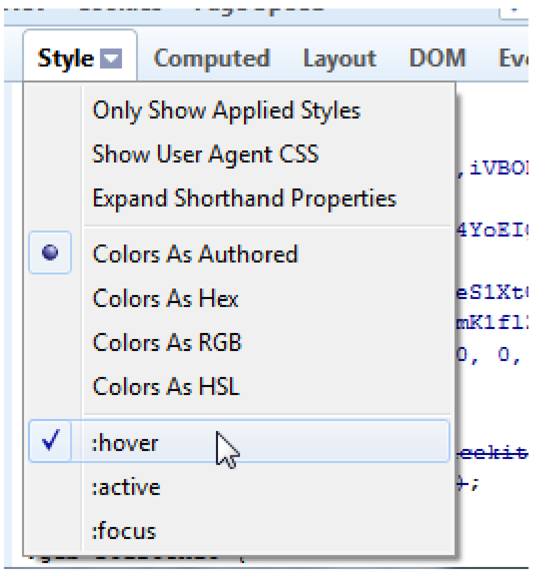
3. In Firebug, you can see the CSS rules responsible for styling the button:



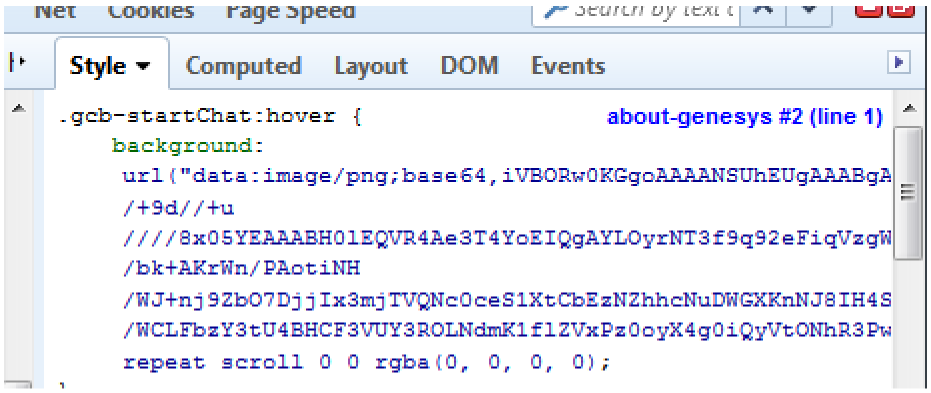
```
<script id="" type="text/javascript">
<script id="" type="text/javascript">
<div class="gcb-startBtnsContainer
gcb-hasChatBtn gcb-hasCobrowseBtn">
    <div class="gcb-startChat"> </div>
    <div class="gcb-startCobrowse"> </div>
</div>
<script id="" type="text/javascript">
<div id="dialog-
confirm" class="wedialog" title="Chat" style
="display: block; z-index: 2147483647;
position: fixed; float: left; top: 286px;
left: 521px;">
</body>
</html>
```

```
.gcb-startChat {                                   about-genesys #2 (line 1)
    background:
     url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgA
     /szHzu0Ynw1pby3KT04bH25r7468v58Nj79eX9+vL
     ///+hSEsKAAABH01EQVR4Ae2TwW5EIQhFeagiKvD
     /Xzt9NbWNA100k+7mxITFDV5QyPDmN65rBXSC8JKVvLDDQa0y6g3ZmY
     /QnEeHpKKaICBxixqpC6eQ2qI6857cm+xyYyZDwG3KSt4cbRGY48aZt
     /YVCQ5o30CWDmHMn7f0sxONj4zvjFNomrdHOQQUbRmxsA04wWmL7v8j
     /WKpKsMVcZQqhLNdma14asisdh+9vynf6xQarScbCjhflx4n8WR4Xb0
     repeat scroll 0 0 rgba(0, 0, 0, 0);
    cursor: pointer;
    height: 83px;
    width: 24px;
}
Inherited from body
```

4. Also check :hover rules to see if there are any rules applied to the button when the mouse cursor is over it. To do this, enable the `:hover` modifier in Firebug:



You can see that there is an additional :hover rule for the button:

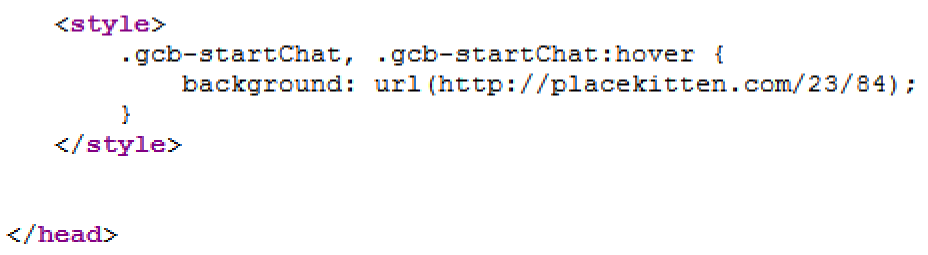


```
.gcb-startChat:hover {                             about-genesys #2 (line 1)
    background:
     url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgA
     /+9d//+u
     ////8x05YEAAABH01EQVR4Ae3T4YoEIQgAYLOyrNT3f9q92eFiqVzgW
     /bk+AKrWn/PAotiNH
     /WJ+nj9Zb07DjjIx3mjTVQNc0ceS1XtCbEzNZhhcNuDWGXKnNJ8IH4S
     /WCLFbzY3tU4BHCF3VUY3ROLNdmK1f1ZVxPz0oyX4g0iQyVtONhR3Pw
     repeat scroll 0 0 rgba(0, 0, 0, 0);
```

5. Base on the rules above, you can prepare your own rules that override the defaults. In this example, we will prepare a custom image with a similar size to the default button and override the background. We will use a public service to generate an image of a kitten via an http request. Here is our CSS:

```
/* 1. copy-paste the selector of the element to override */
 .gcb-startChat, .gcb-startChat:hover {
 /* 2. override some rules: */
   background: url(http://placekitten.com/23/84);
 }
```

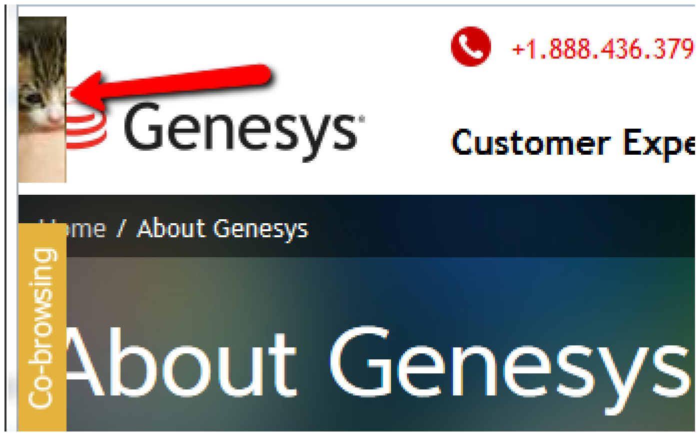
6. Now we add our CSS to the <head> section of our site. The way this is done will depend on the technology your website uses. We want to add code like this to the webpage:



```
    <style>
        .gcb-startChat, .gcb-startChat:hover {
            background: url(http://placekitten.com/23/84);
        }
    </style>


 </head>
```

7. Reload the page. Now, the Live Chat button is replaced with our new image:



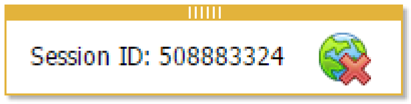
> **Tip**
>
> The `Co-browsing` button can be customized in the same way.

End of Procedure

Example: Customizing the Toolbar Position

Customization Type: **JavaScript Based**

In this example, we will customize the starting position of the Co-browse toolbar:



Prerequisites

1. Experience with JavaScript, jQuery, and browser developer tools such as Firebug.
2. This example uses the the jQuery library to work with the DOM and assumes that jQuery is available on the page.

Start of Procedure

1. Obtain the Co-browse API. See Accessing the Co-browse and Chat APIs for more information about obtaining the Co-browse API. In this example, we use the single-function mechanism.

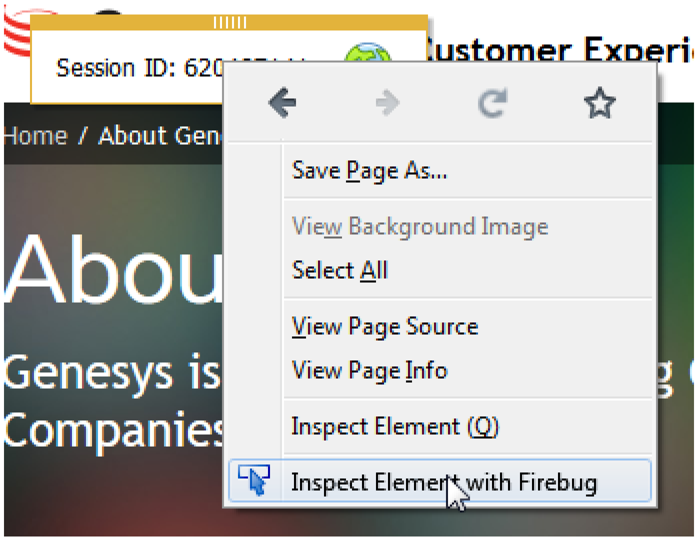
```
var _genesys = {
    cobrowse: {
        onReady: function(cobrowseAPI) {
            // ...
        }
    }
}
```

2. Use the Co-browse API to subscribe to the `onSessionSarted` Co-browse event.

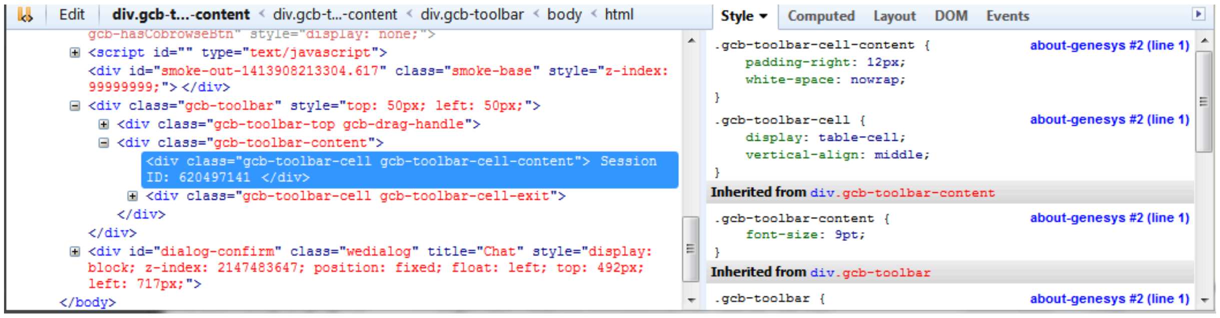
```
cobrowseAPI.onSessionStarted.add(function() {
    // ...
});
```

> **Tip**
> You can subscribe to the readiness of the UI using the onReady Callback.

3. Use Firebug to determine which element to customize:
   a. Start a Co-browse session.
   b. Right click the Co-browse toolbar and click `Inspect Element with Firebug` to open the Firebug tool.



c. In Firebug, you can see that the element selected for inspection is not the toolbar itself but one of its sub-elements.

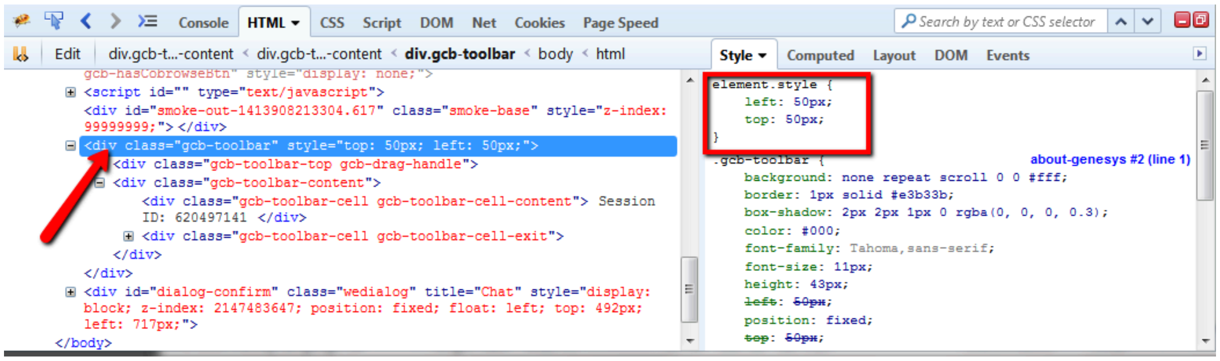

Edit
div.gcb-t...-content < div.gcb-t...-content < div.gcb-toolbar < body < html
Style
Computed
Layout
DOM
Events

```
gcb-hasCobrowseBtn" style="display: none;">
⊞ <script id="" type="text/javascript">
  <div id="smoke-out-1413908213304.617" class="smoke-base" style="z-index:
  99999999;"> </div>
⊟ <div class="gcb-toolbar" style="top: 50px; left: 50px;">
  ⊞ <div class="gcb-toolbar-top gcb-drag-handle">
  ⊟ <div class="gcb-toolbar-content">
      <div class="gcb-toolbar-cell gcb-toolbar-cell-content"> Session
      ID: 620497141 </div>
    ⊞ <div class="gcb-toolbar-cell gcb-toolbar-cell-exit">
    </div>
  </div>
⊞ <div id="dialog-confirm" class="wedialog" title="Chat" style="display:
  block; z-index: 2147483647; position: fixed; float: left; top: 492px;
  left: 717px;">
</body>
```

```
.gcb-toolbar-cell-content {                 about-genesys #2 (line 1)
    padding-right: 12px;
    white-space: nowrap;
}
.gcb-toolbar-cell {                         about-genesys #2 (line 1)
    display: table-cell;
    vertical-align: middle;
}
Inherited from div.gcb-toolbar-content
.gcb-toolbar-content {                      about-genesys #2 (line 1)
    font-size: 9pt;
}
Inherited from div.gcb-toolbar
.gcb-toolbar {                              about-genesys #2 (line 1)
```

You can also see from the `Styles` tab that position is not set for this element.

d. You need to find the element whose position you want to change. From the DOM tree in Firebug you can see that this is the `.gcb-toolbar`.


Console
HTML
CSS
Script
DOM
Net
Cookies
Page Speed
Search by text or CSS selector
Edit
div.gcb-t...-content < div.gcb-t...-content < div.gcb-toolbar < body < html
Style
Computed
Layout
DOM
Events

```
gcb-hasCobrowseBtn" style="display: none;">
⊞ <script id="" type="text/javascript">
  <div id="smoke-out-1413908213304.617" class="smoke-base" style="z-index:
  99999999;"> </div>
⊟ <div class="gcb-toolbar" style="top: 50px; left: 50px;">
    <div class="gcb-toolbar-top gcb-drag-handle">
  ⊟ <div class="gcb-toolbar-content">
      <div class="gcb-toolbar-cell gcb-toolbar-cell-content"> Session
      ID: 620497141 </div>
    ⊞ <div class="gcb-toolbar-cell gcb-toolbar-cell-exit">
    </div>
  </div>
⊞ <div id="dialog-confirm" class="wedialog" title="Chat" style="display:
  block; z-index: 2147483647; position: fixed; float: left; top: 492px;
  left: 717px;">
</body>
```

```
element.style {
    left: 50px;
    top: 50px;
}
.gcb-toolbar {                              about-genesys #2 (line 1)
    background: none repeat scroll 0 0 #fff;
    border: 1px solid #e3b33b;
    box-shadow: 2px 2px 1px 0 rgba(0, 0, 0, 0.3);
    color: #000;
    font-family: Tahoma,sans-serif;
    font-size: 11px;
    height: 43px;
    ~~left: 50px~~;
    position: fixed;
    ~~top: 50px~~;
```

4. Now, we will use jQuery to override the position of the toolbar. We will set the starting position to 100 pixels from the top edge and 300 pixels from the left edge.

```
jQuery('.gcb-toolbar').css({
   top: 100,
   left: 300
});
```

5. Putting it all together, we have the following:

```
// 1. Get the API
var _genesys = {
    cobrowse: {
        onReady: function(cobrowseAPI) {

            // 2. Use the API to subscribe on “session started” event
            cobrowseAPI.onSessionStarted.add(function() {

                // 3. Use jQuery to get the toolbar and reposition it
                jQuery('.gcb-toolbar').css({
                    top: 100,
                    left: 300
                });

            });

        }
    }
};
```

End of Procedure

Tip

You can achieve similar results using CSS:

```
.gcb-toolbar {
    top: 100px !important;
    left: 300px !important;
}
```

This solution is less ideal because you would have to use the `!important` modifier to override the inline styles of the element and this modifier would make the toolbar non-draggable.

Localization

Genesys Co-browse localization is split into three parts:

- Localizing the Customer UI
- Localizaing the Built-in Chat Widget UI
- Localizing the Agent UI

Co-browse is localized for English by default. To modify the default English localization or add localization for other languages, see the following sections on this page:

- Localizing the Customer Co-browse and Chat Widget UIs
- Localizing the "Live Chat" and "Co-browsing" buttons
- Localizing the agent UI
- Caching and updating l10n files
- Built-in localization

Localizing the Customer Co-browse and Chat Widget UIs

To localize the customer Co-browse and Chat Widget UIs, configure the `localization` option in the `chat` and `cobrowse` subsections of the global configuration variable. By default, `_genesys` is the global configuration variable. To configure the `localization` option, do one of the following:

- Pass localization data to the configuration object in you website instrumentation. You can provide localization directly as a JS object or provide localization using a function.
- Save the localization in **.json** files and serve them as JSONP. The Co-browse application uses these files on top of the default built-in values.

  The localization files are plain JSON files, loaded through a JSONP request. This means they can be hosted on any domain as long as JSONP is supported by the hosting server. Co-browse uses a standard `callback` argument for the callback function name. See Serving JSONP for information on how to serve JSONP using Co-browse servers.

Provide Localization Directly as a JS Object

The simplest way to provide localization is to use a plain JavaScript object of key value pairs.

Example:

```
<script>
_genesys = {
    chat: {
        localization: {
            chatTitle: "The cozy chat"
        }
    }
};
</script>
```

Provide Localization Using a Function

If a JavaScript object is not sufficient, you can use a function to provide localization. For example, you can use a function to figure out localization at runtime.

Example:

```
<script>
_genesys = {
    cobrowse: {
        localization: function() {
            if (window.currentLanguage = 'ru') {
                return { toolbarContent: "Ключ сессии: {sessionId}" }
            }
        }
    }
};
</script>
```

Asynchronous Function

Your function can also be asynchronous and you can use an asynchronous function to load localization how you prefer. To tell the configuration variable that your function is asynchronous, create your function with an argument. Customarily, the argument is done. We will pass a function that you have to call whenever the localization loads.

Example:

```
<script>
_genesys = {
    cobrowse: {
        localization: function(done) {
            jQuery.get('my/cobrowse/localization.json', done);
        }
    }
};
</script>
```

Warning

Functions you pass will be called after our scripts initialize, shortly after the `document ready` event. If you use an asynchronous function, initialization will be postponed until you call the done function.

Provide Localization Using an External JSON File

If you pass a string to the `localization` configuration option, the string will be treated as a URL to an external JSON file.

Example:

```
<script>
_genesys = {
    cobrowse: {
        localization: '//example.com/cobrowse-l10n/2014-10-08/cobrowse-fr.json'
    },
    chat: {
        localization: '//another-example.com/chat-l10n/2014-10-08/chat-fr.json'
    }
};
</script>
```

Important

Our scripts will attempt to load localization files using JSONP. You can use Co-browse server to serve files as JSONP for you. See Serving JSONP.

If the files can not be loaded, initialization of our functionality will be blocked.

You must save localization JSON files with UTF-8 encoding.

Localizing the "Live Chat" and "Co-browsing" buttons

The "Live Chat" and "Co-browsing" default buttons are images and localization changes will not affect the text on these buttons. Instead, you can localize the buttons using one of the following methods:

1. Provide custom localized buttons instead of the defaults. See Integrated JavaScript Application#Providing Custom HTML for Buttons for details.
2. Override how the buttons look using CSS. See Customizing the CSS for details.

Localizing the Agent UI

The agent UI is localized by updating the `localization` configuration option to the URL of the JSON localization file in the slave section of the Co-browse Server application. Unlike the customer UI, the value is changed in the Genesys Configuration server and not via the instrumentation script. The agent UI can not be configured by passing a JavaScript object or function.

Important

The localization file is loaded using JSONP. You can use Co-browse server to serve files as JSONP for you or configure your own infrastructure for JSONP support. See Serving JSONP.

Caching and updating Localization (l10n) files

Requests to the l10n files are made on every page that is instrumented with Co-browse, before the Co-browse UI is displayed. Genesys recommends that you implement a caching mechanism for these files if you host them on your servers.

For best performance, add far-future expiration headers (for example, `Expires`, `Cache-Control` or both) to your l10n files. This prevents the browser from requesting these files on each page. Instead, it will take them from the cache. This reduces the start-up time for the Co-browse UI and cuts down traffic for the end user. If you modify a localization file with a far-future expiration header, the browser must request the new version of the file from the server instead of taking it from the cache. To force the browser to do this you must change the URL of the file. You can do this by updating the corresponding `localization` parameter in Co-browse instrumentation after either putting the modified file in a new directory or updating the file's name.

For example, consider the case where you have set up your own server to host the Co-browse localization files for the customer UI:

```
<script>
var _genesys = {
    cobrowse: {
        localization: '//example.com/cobrowse-l10n/2014-09-07/cobrowse-fr.json'
    }
};
</script>
<COBROWSE_INSTRUMENTATION_SCRIPT>
```

Next, you modify some of your localization files and want them to be refreshed for all users, so you create a new directory and update your Co-browse instrumentation:

```
<script>
var _genesys = {
    cobrowse: {
        localization: '//example.com/cobrowse-l10n/2014-10-08/cobrowse-fr.json'
    }
};
</script>
<COBROWSE_INSTRUMENTATION_SCRIPT>
```

Now, any browsers that had cached the files will be reload the files and re-cache them.

Built-in localization

This section lists the default localization values and keys.

You can use the code snippets in this section to create your own localization files. To do so, copy and save the code snippet as a `.json` file.

Important

You must save your localization JSON files using UTF-8 encoding.

Tip

It is not necessary to list **all** the keys in your localization JSON file. In your JSON file, you can specify only the keys you wish to override. Any key not specified will use the default localization value.

Customer UI

```
{
  "agentJoined": "Representative has joined the session",
  "youLeft": "You have left the session. Co-browse is now terminated.",
  "sessionTimedOut": "Session timed out. Co-browse is now terminated.",
  "sessionInactiveTimedOut": "Session timed out. Co-browse is now terminated.",
  "agentLeft": "Representative has left the session. Co-browse is now terminated.",
  "sessionError": "Unexpected error occurred. Co-browse is now terminated.",
  "sessionsOverLimit": "Representative is currently busy with another Co-browse session. Co-
browse is now terminated.",
  "serverUnavailable": "Could not reach Co-browse server. Co-browse is now terminated.",
  "sessionStarted": "Your co-browse session ID is {sessionId}. Please spell it to our
representative to continue with co-browsing.",
  "navRefresh": "Representative has refreshed the page. Reloading.",
  "navBack": "Representative has pressed the \"Back\" button. Reloading page.",
  "navForward": "Representative has pressed the \"Forward\" button. Reloading page.",
  "navUrl": "Representative has requested navigation. Reloading page.",
  "navFailed": "Navigation request by representative has failed.",
  "toolbarContent": "Session ID: {sessionId}",
  "contentMasked": "Content is hidden from representative",
  "contentMaskedPartially": "Some content is hidden from representative",
  "exitBtnTitle": "Exit Co-browse session",
  "areYouOnPhone": "Are you on the phone with our representative?",
  "areYouOnPhoneOrChat": "Are you on the phone or chat with our representative?",
  "connectBeforeCobrowse": "You need to be connected with our representative to continue with
co-browsing. Please call us or start a live chat with us, and then start Co-browse again.",
  "sessionStartedAutoConnect": "Co-browse session started. Waiting for representative to
connect to the session…",
  "browserUnsupported": "Unfortunately, your browser is not currently supported.<br><br>
Supported browsers are: <ul><li><a target='_blank' href='http://www.google.com/chrome'>Google
Chrome</a></li><li><a target='_blank' href='http://www.firefox.com/'>Mozilla
Firefox</a></li><li><a target='_blank' href='http://microsoft.com/ie'>Internet Explorer 9 and
above</a></li><li><a target='_blank' href='https://www.apple.com/safari'>Safari 6 and
above</a></li></ul>",
  "chatIsAlreadyRunning": "Chat is already running on another page.",
  "modalTitle": "Co-browse",
  "modalYes": "Yes",
  "modalNo": "No",
  "writeModeInProgress": "Agent has control over the page.",
  "downgradeMode": "Revoke control",
  "modeUpgraded": "Co-browse session was upgraded. Agent has control over the page.",
  "modeDowngraded": "Co-browse session was downgraded. Agent has no control",
  "modeUpgradeRequested": "Agent requests upgrading Co-browse session to \"write\" mode. In
\"write\" mode agent will have control over the page."
}
```

Important

The `modalTitle` and `serverUnavailable` keys were added to Genesys Co-browse in release 8.5.001.xx.

Chat Widget

See Localization in the Chat Widget JS API.

Agent

```
{
  "invalidSessionID": "Session ID is invalid or has expired.",
  "navRefresh": "Refresh is pressed. Reloading page.",
  "navBack": "Back is pressed. Reloading page.",
  "navForward": "Forward is pressed. Reloading page.",
  "youLeft": "You have left the session. Co-browse is now terminated.",
  "customerLeft": "Customer has left the session. Co-browse is now terminated.",
  "exitBtnText": "Exit Session",
  "sessionIdText": "Session ID: {sessionId}",
  "enterSessionIdText": "Session ID:",
  "modeWrite": "Mode: Write",
  "modePointer": "Mode: Pointer",
  "downgradeToPointer": "Downgrade to pointer mode",
  "navigationDenied": "Navigation request has failed.",
  "maskedNodeTitle": "This content is visible only to customer",
  "unsupportedNodeTitle": "Some data is missing due to Co-browse limitations",
  "navigationRestricted": "Navigation is restricted in Pointer mode.",
  "modeUpgraded": "Switched to Write mode. Now you can interact with the page.",
  "modeDowngraded": "Switched to Pointer mode. Customer can only see your cursor and clicks.",
  "writeModeRequested": "Requested Write mode. Waiting for customer approval.",
  "modeUpgradeDenied": "Customer declined upgrading to Write mode."
}
```

Attached Data Overview

Co-browse does not have a dedicated type of interaction. Data for reporting must be attached to the primary chat or voice interaction. Almost all data necessary is automatically attached by IWS/WDE plug-ins as described below. The only default Co-browse statistic not automatically attached to the interaction is `source/web` (key/value).

Important

Please note the following about attached data:

- **CoBrowseDuration** is not used for reporting purposes.
- **CoBrowseEndTime** was removed from Reporting 8.5.0.

Manually Attached Data

Data for custom statistics should be attached on the browser side with the creation of a primary interaction.

Automatically Attached Data

The table below describes the Co-browse attached data keys. The section Events that Update Co-browse Attached Data describes how the data is updated throughout a co-browse session.

Key	Interaction	Data Type	Attached By	Used In	Value
source	chat	String	Browser	Reporting	Source of the interaction. Values: `web` (desktop and mobile browsers) or `mobile` (apps) Currently, the value is always `web`.
referrer	chat	String	Browser	Reporting	URL of the previous page (`document.referrer`). This value is used to detect what website the customer comes from.

Key	Interaction	Data Type	Attached By	Used In	Value
					The value might be shortened.
url	chat	String	Browser	Reporting	URL of the page where the chat started (`document.location`). The value might be shortened.
pageTitle	chat	String	Browser	Reporting	Title of the page where the chat started (`document.title`).
CoBrowseAssociated	co-browse on chat or voice	String	GCB plug-in for WDE	Reporting	Values: • `Yes`—This key is appended to the primary interaction with a value of `Yes` when an agent joins their first Co-browse session. The value is set to `Yes` each time a new Co-browse session begins within the same chat. • `No`—The value is set to `No` when an agent exits a Co-browse session.
CoBrowseSessionsQuantity	Co-browse on chat or voice	Integer	GCB plug-in for WDE	Reporting	Value is equal to the number of Co-browse sessions for the current interaction.

Key	Interaction	Data Type	Attached By	Used In	Value
					If there is more than one Co-browse session for the primary interaction, the value is increased by one with each new Co-browse session.
CoBrowseSessionId	Co-browse on chat or voice	String	GCB plug-in for WDE	Reporting	The value is the unique Co-browse session ID.
CoBrowseStartTime	Co-browse on chat or voice	String	GCB plug-in for WDE	Reporting	An epoch of time specified in milliseconds. It is the start of the first Co-browse session for the current primary interaction.
IsCoBrowseDenied (New in 8.5.003)	Co-browse on chat or voice	String	GCB plug-in for WDE	Pulse Reporting	Available values are `Yes` or `No`. • If the Co-browse session is **not** allowed to start, **`IsCoBrowseDenied`**=`Yes` is attached if absent or set to `Yes` if already attached. • If the Co-browse session is allowed to start, **`IsCoBrowseDenied`**=`No` is set only if **`IsCoBrowseDenied`** is already attached. If **`IsCoBrowseDenied`** is absent, it

Key	Interaction	Data Type	Attached By	Used In	Value
					will remain absent.

Events that Update Co-browse Attached Data

This section describes how the attached data is updated throughout a co-browse session.

Agent Joins the First Co-browse Session

The WDE Co-browse plug-in appends the following Co-browse session data to the primary Chat or Voice interaction when an agent joins a session:

Attached Data	Data Type	Value
CoBrowseAssociated	String	The initial value is set to Yes.
CoBrowseSessionsQuantity	Integer	The number of Co-browse sessions for the current interaction.
CoBrowseSessionId	String	The unique session ID.
CoBrowseStartTime	String	An epoch of time specified in milliseconds. It matches the start of the first Co-browse session for the current interaction.

Agent Exits

When an agent exits a Co-browse session, the WDE Co-browse plug-in updates the following data:

Attached Data	Data Type	Value
CoBrowseAssociated	String	Value is changed to No.
CoBrowseDuration	Integer	The duration in seconds for the Co-browse session. Summarized the duration of all Co-browse sessions for the current interaction. This data is **not** used in Reporting.
CoBrowseEndTime	String	An epoch time specified in milliseconds. The epoch time corresponds to the end of the last Co-browse session for the current interaction. This data is **not** used in Reporting 8.5.0.

Agent Joins More Than One Co-browse Session

If there is more than one Co-browse session for the primary interaction, the WDE Co-browse plug-in changes the following attached data with each new Co-browse session:

Attached Data	Data Type	Value
CoBrowseAssociated	String	Value is updated to `Yes`.
CoBrowseSessionsQuantity	Integer	The number of Co-browse sessions for the current interaction.
CoBrowseSessionId	String	The unique session ID.

Co-browse Simultaneous Session Limits

Starting with Genesys Co-browse release 8.5.003.04, you can enable one-session limitations and configure the number of simultaneous co-browsing sessions an agent can participate in with the `agentSessionsLimit` option in the `cobrowse` section of the Workspace Desktop Edition application.

- If the Co-browse session is **not** allowed to start, **`IsCoBrowseDenied`**=Yes is attached if absent or set to `Yes` if already attached.
- If the Co-browse session is allowed to start, **`IsCoBrowseDenied`**=No is set only if **`IsCoBrowseDenied`** is already attached. If **`IsCoBrowseDenied`** is absent, it will remain absent.

Using Attached Data for Reporting

Almost all attached data is used in the statistics and filters for Stat Server Application configuration. You can import configuration option profiles for either Pulse or CCPulse+ reporting:

- Pulse: **<Genesys Co-browse Sample Reporting Templates root>/Pulse/GCB/StatProfile_Pulse.cfg**
- CCPulse+: **<Genesys Co-browse Sample Reporting Templates root>/CCPulse+/StatProfile_CCPulse.cfg**

The configuration option profiles contain the following statistics and filters:

Pulse Statistics and Filters

These statistics and filters are specified in the *current* and *total* type templates located in `/Pulse/GCB/current` and `/Pulse/GCB/total` folders.

For details, see Pulse Templates.

[+] Click to View Pulse Statistics and Filters

Statistics

```
[Agents_CurrentNumber]
Category=CurrentNumber
Description=Current number of agents working with interactions
MainMask=CallInternal,CallConsult,CallInbound
Objects=GroupAgents,GroupPlaces,Tenant
Subject=DNStatus

[Agents_CurrentNumber_Inbound]
Category=CurrentNumber
Description=Current number of agents working with inbound interactions
MainMask=CallInbound
Objects=GroupAgents,GroupPlaces,Tenant
Subject=DNStatus

[Agents_MaxNumber]
Category=MaxNumber
Description=Max number of agents worked with interactions
MainMask=CallInternal,CallConsult,CallInbound
Objects=GroupAgents,GroupPlaces,Tenant
Subject=DNStatus
```

```
[Agents_MaxNumber_Inbound]
Category=MaxNumber
Description=Max number of agents worked with inbound interactions
MainMask=CallInbound
Objects=GroupAgents,GroupPlaces,Tenant
Subject=DNStatus

[CurrentExAgentState]
Category=CurrentState
MainMask=*
Objects=Agent
Subject=DNAction

[Interactions_CurrentHandling]
Category=CurrentNumber
Description=Current number of interactions handling
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place,Tenant
Subject=DNAction

[Interactions_Current_Inbound]
Category=CurrentNumber
Description=Current number of inbound interactions handling
MainMask=CallInbound
Objects=Agent,GroupAgents,GroupPlaces,Place,Tenant
Subject=DNAction

[Interactions_TotalHandled]
Category=TotalNumber
Description=Total number of interactions handled
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place,Tenant
Subject=DNAction

[Interactions_Total_Inbound]
Category=TotalNumber
Description=Total number of inbound interactions handled
MainMask=CallInbound
Objects=Agent,GroupAgents,GroupPlaces,Place,Tenant
Subject=DNAction

[Interactions_TotalDuration]
Category=TotalTime
Description=Total time of interactions handled
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place,Tenant
Subject=DNAction
```

Filters

```
[Filters]
```

Chat_Co-browse=PairExists("CoBrowseAssociated","*") & PairExists("MediaType","chat")
Chat_Co-browseAlive=PairExists("CoBrowseAssociated","Yes") & PairExists("MediaType","chat")
ChatInteraction=PairExists("MediaType","chat")
Voice_Co-browse=PairExists("CoBrowseAssociated","*") & (PairExists("MediaType","voice") | (MediaType=voice))
Voice_Co-browseAlive=PairExists("CoBrowseAssociated","Yes") & (PairExists("MediaType","voice") | (MediaType=voice))
VoiceCallInteraction=PairExist("MediaType", "voice") | (MediaType=voice)
WebChat_Co-browse=PairExists("source","web") & PairExists("CoBrowseAssociated","*") & PairExists("MediaType","chat")
WebChat_Co-browseAlive=PairExists("source","web") & PairExists("CoBrowseAssociated","Yes") & PairExists("MediaType","chat")
WebChatInteraction=PairExists("source","web") & PairExists("MediaType","chat")CoBrowseAssociated=PairExists("CoBrowseAssociated","*")
CoBrowseSessionId=PairExists("CoBrowseSessionId","*")
CoBrowseStartTime=PairExists("CoBrowseStartTime","*")
CoBrowseSessionsQuantity=PairExists("CoBrowseSessionsQuantity","*")

Common Chat Templates

Common chat templates for Pulse the configuration file with statistics and filters for the Stat Server application are located in the `/Pulse/common` folder.

CCPulse+ and DMA Reporting

These statistics and filters are specified in the real-time and historical templates located in the `/CCPulse+/runtime` and `/CCPulse+/historical` folders.

These statistics and filters are specified in the real-time and historical templates located in the "runtime" and "historical" folders. For details, see CCPulse+ Templates.

[+] Click to View CCPulse+ Statistics and Filters

Statistics

[Agents_CurrentNumber]
Category=CurrentNumber
Description=Current number of agents working with interactions
MainMask=CallInternal,CallConsult,CallInbound
Objects=GroupAgents,GroupPlaces
Subject=DNStatus

[Agents_CurrentNumber_Inbound]
Category=CurrentNumber
Description=Current number of agents working with inbound interactions
MainMask=CallInbound

```
Objects=GroupAgents,GroupPlaces
Subject=DNStatus

[Agents_MaxNumber]
Category=MaxNumber
Description=Max number of agents worked with interactions
MainMask=CallInternal,CallConsult,CallInbound
Objects=GroupAgents,GroupPlaces
Subject=DNStatus

[Agents_MaxNumber_Inbound]
Category=MaxNumber
Description=Max number of agents worked with inbound interactions
MainMask=CallInbound
Objects=GroupAgents,GroupPlaces
Subject=DNStatus

[CurrentExAgentState]
Category=CurrentState
MainMask=*
Objects=Agent
Subject=DNAction

[Interactions_CurrentHandling]
Category=CurrentNumber
Description=Current number of interactions handling
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place
Subject=DNAction

[Interactions_Current_Inbound]
Category=CurrentNumber
Description=Current number of inbound interactions handling
MainMask=CallInbound
Objects=Agent,GroupAgents,GroupPlaces,Place
Subject=DNAction

[Interactions_TotalHandled]
Category=TotalNumber
Description=Total number of interactions handled
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place
Subject=DNAction

[Interactions_Total_Inbound]
Category=TotalNumber
Description=Total number of inbound interactions handled
MainMask=CallInbound
Objects=Agent,GroupAgents,GroupPlaces,Place
Subject=DNAction

[Interactions_TotalDuration]
Category=TotalTime
Description=Total time of interactions handled
```

```
MainMask=InteractionHandling
Objects=Agent,GroupAgents,GroupPlaces,Place
Subject=DNAction
```

Filters

```
[Filters]
Chat_Co-browse=PairExists("CoBrowseAssociated","*") & PairExists("MediaType","chat")
Chat_Co-browseAlive=PairExists("CoBrowseAssociated","Yes") & PairExists("MediaType","chat")
ChatInteraction=PairExists("MediaType","chat")
Co-browse=PairExists("CoBrowseAssociated","*")
Co-browseAlive=PairExists("CoBrowseAssociated","Yes")
VoiceCallInteraction=PairExist("MediaType", "voice") | (MediaType=voice)
Voice_Co-browse=PairExists("CoBrowseAssociated","*") & (PairExists("MediaType","voice") |
(MediaType=voice))
Voice_Co-browseAlive=PairExists("CoBrowseAssociated","Yes") & (PairExists("MediaType","voice") |
(MediaType=voice))
CoBrowseSessionId=PairExists("CoBrowseSessionId","*")
CoBrowseStartTime=PairExists("CoBrowseStartTime","*")
CoBrowseSessionsQuantity=PairExists("CoBrowseSessionsQuantity","*")
```

Currently, the following attached data is not used for Reporting purposes:

- CoBrowseDuration

You can customize the Co-browse Reporting Templates and the statistics and filters to suit your needs. You can also create your statistics and filters using Co-browse-related data or any other data attached to the Primary interaction. The statistics and filters you create can then be used in the existing reporting templates or you can create your own templates. For details, refer to the following documents:

- Stat Server 8.1 User's Guide
- Reporting 8.0 CCPulse+ Administrator's Guide

Using Attached Data to Customize Business Processes

You can use Co-browse-related attached data to organize Business Processes against Chat or Voice interactions. For details, please refer to the following documents:

- Composer Documentation
- eServices 8.1 User’s Guide
- Universal Routing 8.1 Strategy Samples
- Universal Routing 8.1 Business Process User's Guide

Monitoring Overview

Genesys Co-browse added monitoring features in **8.5.001** and extended those features in **8.5.002**.

Monitoring features added in 8.5.001

Starting with **8.5.001**, Co-browse Server integrates with a Metrics Java library to allow you to generate Co-browse reporting metrics about Co-browse Server performance and the processing of Co-browse sessions. While the metrics library gives several ways to report on current values, only reporting through the JMX JConsole interface is available in **8.5.001**.

Monitoring features extended in 8.5.002

Genesys Co-browse **8.5.002** extends monitoring functionality:

- New *logging reporter* lets you report metric values in the console or log.
- Co-browse Server can now generate *Monitoring Alarms* you can use to improve Co-browse performance.

See the links below for more on the reports and metrics available through Co-browse Server.

Monitoring Co-browse

In **8.5.002+**, Genesys Co-browse Server's monitoring functionality generates the following:

- Co-browse Performance Counters (KPIs) using:
  - JMX reporter, observable with any JMX interface
  - Logging reporter for metrics
- Co-browse Monitoring Alarms using:
  - Logging reporter for alarms
  - Message Server reporter

Performance Counters (KPIs)

This article describes the basics of how Co-browse works together with the third-party Metrics Java library to provide reporting metrics about your server's Co-browse sessions. It also gives a walk-through of how to set up a sample JMX interface in order to view the metrics the Co-browse Server creates.

Co-browse and Metrics Library

Genesys Co-browse integrates with the third-party Metrics Java library, a toolkit that support all kinds of metrics out of the box: for example, counter, timer, histogram, and gauge.

This Metric library gives you several ways to report on current values: JMX (the main method), REST (for performance testing), and Logging.

About Co-browse Metrics

Starting with release 8.5.100.05, the Co-browse Server integrates with the Metrics Library client for the Java Management Extensions (JMX) reporter interface. JMX lets you observe Co-browse metrics using JMX tools.

Co-browse Server 8.5.002 extended metrics functionality to support logging to a file and the console.

Overview of Available Co-browse Metrics

Co-browse Server generates these kinds of metrics:

- Current count of sessions in different states (counter metric)
- Count of completed sessions since the start of the server (counter metric)
- Session timings (timer/histogram)
  - Agent overall rendering time (histogram)
  - Agent stages rendering time (histogram)
  - Co-browse session initialization on server side (timer)
  - Time of Customer, Agent and Controller joining to the Co-browse session (timer)
- Sessions interrupted without accept (counter metric).

Breakdown of Available Co-browse Metrics

Metric name	Description	Added in version:
ActiveSessions	Sessions set to "Activated" status when session is created by Customer and joined by Agent	8.5.001
CanceledInactiveSessions	Sessions canceled by initiator	8.5.001
InactiveSessions	Sessions set to "Inactive" status when session is created by Customer but waiting for Agent to join	8.5.001
LiveSessions	All sessions in statuses "Inactive" or "Activated".	8.5.001
NormallyEndedActiveSessions	Sessions ended during period of two sides Co-browse activity	8.5.001
TerminatedByUserDisconnectionSessions	Sessions ended through User timeout disconnect	8.5.001
TimeoutedInactiveSessions	Sessions ended by timeout in awaiting for Agent connection	8.5.001
TotalFinishedSessions	Total count of all finished sessions	8.5.001
CreateSessionAverage	Histogram showing the timings for session creation on the server side	8.5.001
JoinSessionAverage	Histogram showing the timings for the server join procedure for each member in a Co-browse activity	8.5.001
SlaveInitAverage	Histogram showing the timings for Agent initialization after a page reload with session ID	8.5.001
SlaveGetSessionAverage	Histogram showing the timings for the Agent to obtain the session environment after a page reload with session ID	8.5.001
SlaveHandshakeAverage	Histogram showing the timings for the Agent handshake via CometD after a page reload with session ID	8.5.001
SlaveJoinAverage	Histogram showing the timings for the Agent to join a session after a page reload with session ID	8.5.001
SlavePageDataAverage	Histogram of timing for Agent got page data since page reload with session ID	8.5.001
SlaveRenderAverage	Histogram showing the timings for the Agent to fully render after a page reload with session ID	8.5.001

Metric name	Description	Added in version:
ServerResponseTime	Histogram showing the average timings for the latest N routings of data from customer browser to agent browser, where N is defined by the `ServerResponseTime.slidingWindowSize` option value.	8.5.002

How To Expose Co-browse metrics through the JMX interface

There are many JMX tools that you can use to observe the metrics Co-browse Server creates:

- JConsole tool bundled with Oracle Java (TM)
- EJTools JMX Browser
- Panoptes
- jManage
- MC4J
- Zabbix

Using JConsole to Observe Co-browse Metrics

To use JConsole to view Co-browse metrics:

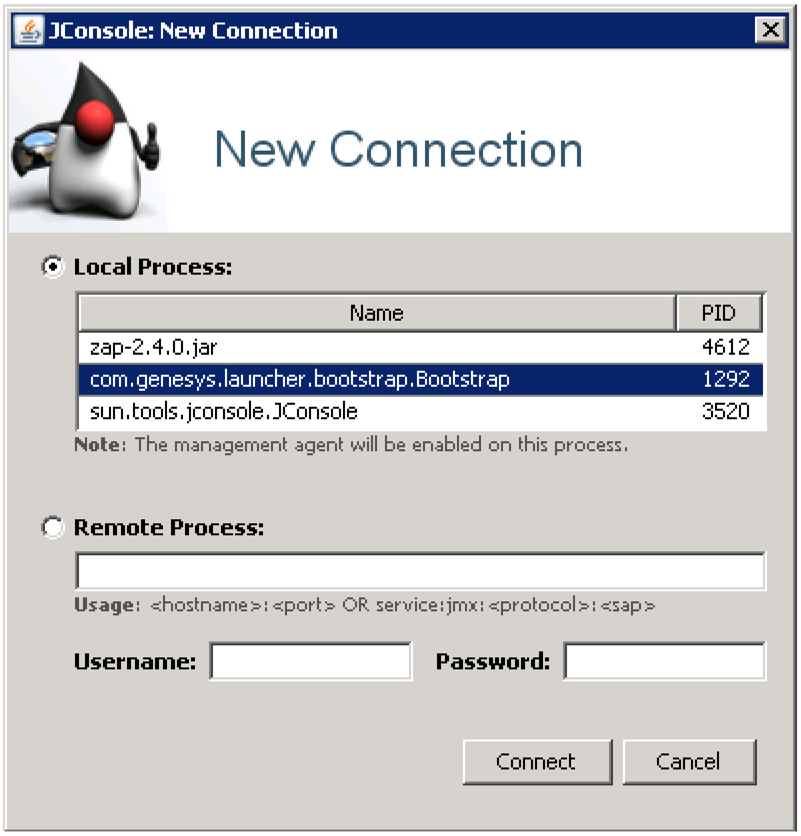
1. Connect JConsole to Co-browse Server
2. Open the JMX panel to view the metrics

Connect JConsole to Co-browse Server

Connecting JConsole to Co-browse Server depends on the Co-browse Server process:

- Connect to Co-browse started as a *local java process*
- Connect to Co-browse started as a *server*
- Connect to Co-browse started as a *Windows service*

Connect to Co-browse started as a *local java process*



(●) **Local Process:**

Name	PID
zap-2.4.0.jar	4612
com.genesys.launcher.bootstrap.Bootstrap	1292
sun.tools.jconsole.JConsole	3520

Note: The management agent will be enabled on this process.

( ) **Remote Process:**

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

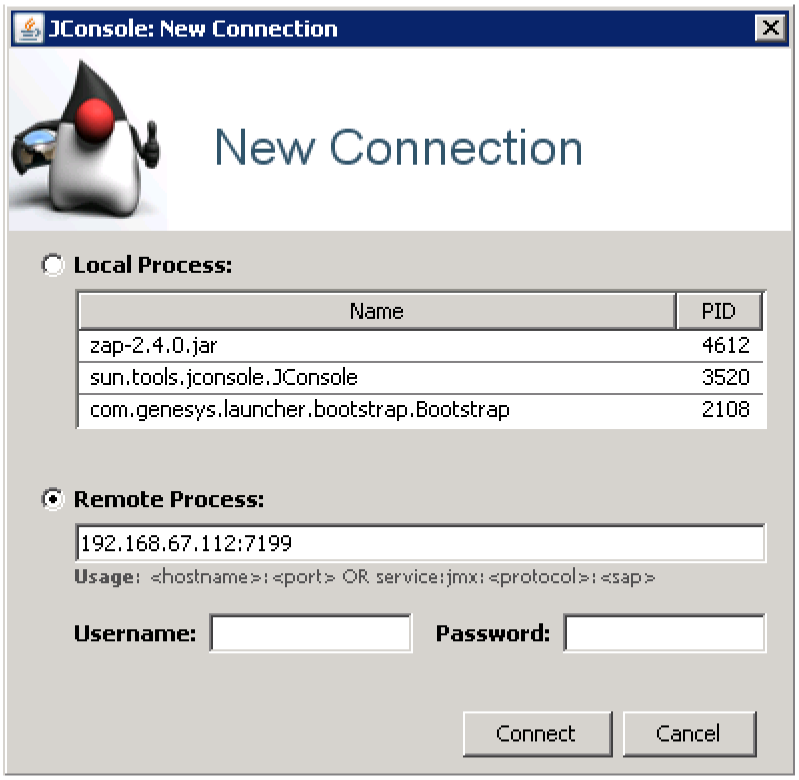
Username: **Password:**

1. Run **jconsole.exe** from the **<jdk>/bin** directory.
2. In the **New Connection** dialog, specify the Co-browse launcher java process.

   If the Co-browse Server was started using a .bat file in the same host where JMX console is opened, specify the following process from the **Local Process** list:

   ```
   com.genesys.launcher.bootstrap.Bootstrap
   ```

Connect to Co-browse started as a *server*



Local Process:

Name	PID
zap-2.4.0.jar	4612
sun.tools.jconsole.JConsole	3520
com.genesys.launcher.bootstrap.Bootstrap	2108

Remote Process: 192.168.67.112:7199

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: Password:

If the Co-browse Server was started remotely as a server, follow these steps:

1. Run **jconsole.exe** from the **<jdk>/bin** directory.
2. Open `setenv.bat` and uncomment all lines under

   ```
   :: Uncomment for enabling JMX Remote. Memorize JMX port.
   ```

   Save your changes.
3. Restart the Co-browse Server application.
4. Specify `host:<JMX port>` in the **Remote Process** section:

Connect to Co-browse started as a *Windows service*

If you started Co-browse Server as a Windows service, first stop the service, reinstall it, and start it again, as follows:

1. Stop the service.
2. Open **setenv.bat** and find the service name in the line set SVC_NAME=
3. Run this command:

   ```
   cobrowse.bat -service SERVICENAME remove
   ```

to remove the service.

4. Open **setenv.bat** and uncomment all lines under

   ```
   :: Uncomment for enabling JMX Remote. Memorize JMX port.
   ```

   Save your changes.
5. Run this command:

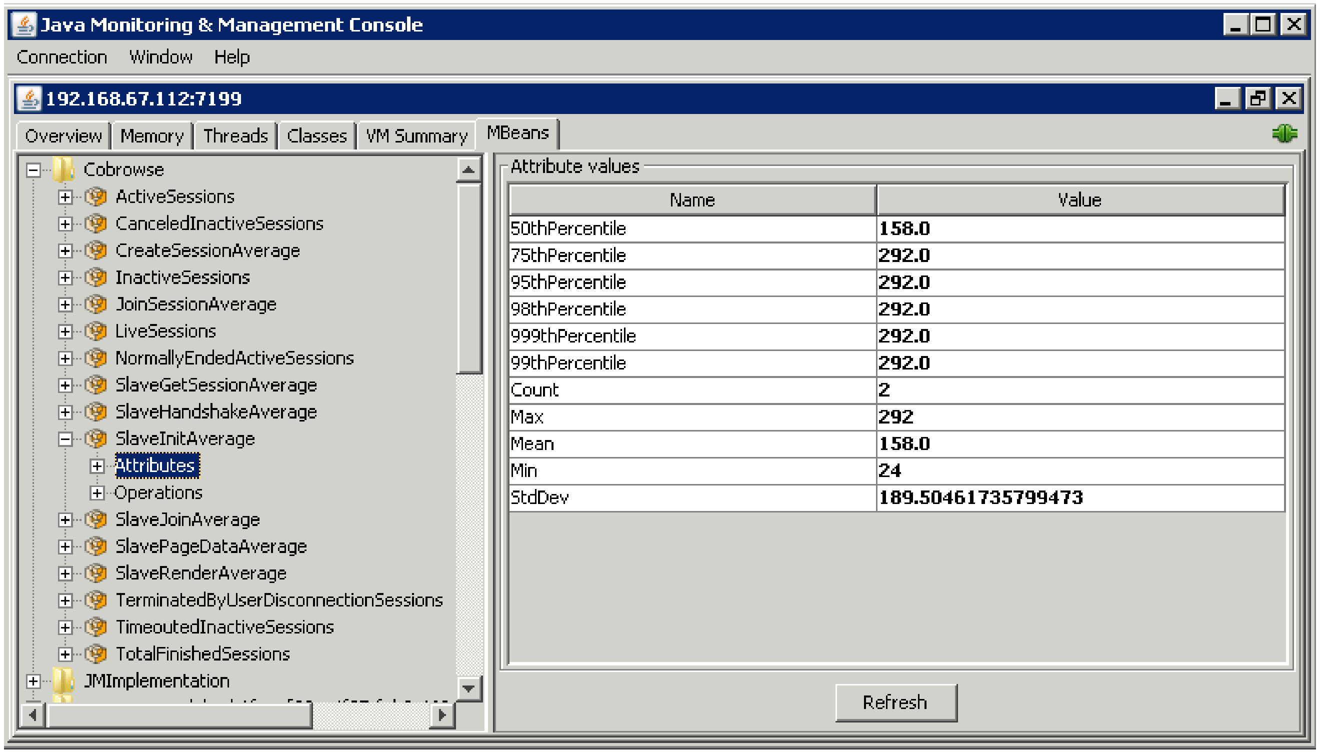
   ```
   cobrowse.bat -service SERVICENAME install
   ```

   to install the service.
6. Start the service.
7. Specify `host:<JMX port>` from the **Remote Process** section, see above.

Once you connect to JConsole to Co-browse Server, you can open the JMX panel to view the metrics.

Open the JMX panel to view the metrics

1. Click `Connect` in the **New Connection** dialog. The JMX panel opens.
2. Open the **MBeans** tab and expand the **Cobrowse** branch. All Co-browse metrics are there.
3. To refresh the metrics, click **Refresh**.



Name	Value
50thPercentile	158.0
75thPercentile	292.0
95thPercentile	292.0
98thPercentile	292.0
999thPercentile	292.0
99thPercentile	292.0
Count	2
Max	292
Mean	158.0
Min	24
StdDev	189.50461735799473

Configuring logging reporter for metrics

Co-browse Server release **8.5.002** extends metric functionality to support logging to a file and to the console.

To configure the logging reporter to log to a file or to the console:

Logging to a file

To enable logging to a *file*:

metrics section

In the **metrics** section of your Co-browse Cluster application configure the following:

- Set **reporter.log.enabled** to `true` (`false` by default)
- Configure **reporter.log.logFrequency** (default value is `30min`)

log section

In the Co-browse Node application **log** section configure the following:

- Set **verbose** to `trace`
- Set `<output>=<log file name>` where output is all, trace, or debug.

Logging to the console

To enable logging to the *console*:

metrics section

In the **metrics** section of you Co-browse Cluster application configure the following:

- Set **reporter.console.enabled** to `true` (`false` by default)
- Configure **reporter.console.logFrequency** (default value is `30min`)

log section

In the Co-browse Node application **log** section configure the following:

- Set **verbose** to `trace`
- Set `<output>=<stdout>` where output is all, trace, or debug.

Monitoring Alarms

Tip

Genesys Co-browse Server **8.5.002** extended metrics functionality by adding monitoring alarms. You can use monitoring alarms to improve Co-browse performance

A *monitoring alarm* is an alert that signals a problem discovered in Co-browse Server. Co-browse Server produces *predefined* and *generic* monitoring alarms.

Predefined monitoring alarms include:

- Heap Memory Usage
- GC Frequency
- GC Latency
- Inactive Sessions
- Jetty Thread Pool Usage
- Server Response Time
- Agent Side Render Latency

The criteria Co-browse Server uses to detect and cancel a problem depend on the monitored metric's specified threshold.

Thresholds

A *threshold* is the basic element used to implement all generated monitoring alarms.

Each threshold is described by the following parameters:

- JMX metric
- Threshold type, predefined or generic
- Related option in the metrics section of the Co-browse Server's configuration
- Log Event ID for `detect` event
- Log Event ID for `cancel` event

Predefined thresholds

Alert generations use predefined thresholds when threshold parameters like `metric`, `Detect Log ID`, and `Cancel Log ID` are predefined and cannot change through configuration.

Generic Thresholds

Generic thresholds let you dynamically set thresholds on any registered metric of type `counter`, `histogram`, or `timer`.

Configuring Monitoring Alarm Reports

You can configure the Logging Reporter and the Message Server Reporter to report monitoring alarms.

Logging Reporter

You can report alarms in the logging subsystem using the *logging reporter*. The logging subsystem is configured in the **log section** of Co-browse Server configuration.

All alarms that detect events are reported in log messages with level **[ERROR]** while all alarms that cancel events have level **[WARN]**.

Detection alarms come in two types:

- fatal alarms with **alarm** log level
- standard alarms with **standard** log level

Cancellation alarms correspond to a **trace** log level.

Message Server Reporter

Starting with release **8.5.002**, Co-browse Server supports a *Message Server reporter* you can use to display alarms in the **Active Alarms** section of Genesys Administrator. By reporting alarms in **Active Alarms**, you simplify application monitoring and avoid detailed logging that can affect system performance.

Configuring Monitoring Alarms

Alarms are log messages reported according to the configured log subsystem. To report a particular alarm in **Active Alarms**, you must configure:

- Message Server Reporter
- Alarm Condition object
- related threshold option in the server application

You can see the dependencies between Alarm Condition objects and related application server configuration options in the Co-browse Alarms Configuration Table.

Important
To apply new Alarm Condition objects, restart Solution Control Server.

Configuring Message Server Reporter

To configure Message Server reporter, specify the following:

1. Message Server Application:

   In the **messages** section, set **db_storage** to `true`.
2. Co-browse Cluster Application:
   1. Add a connection to the Message Server application.
   2. Configure the **metrics** section:
      - Set **reporter.messageServer.enabled** to `true` (default value).
      - Set the **reporter.messageServer.logFrequency**. The default value is 30 minutes.
3. Co-browse Node application's **log** section:
   - Set the **verbose** option to `standard` for only error messages or to `trace` for error and info messages.
   - Set the all, trace, or debug options to value `network`.

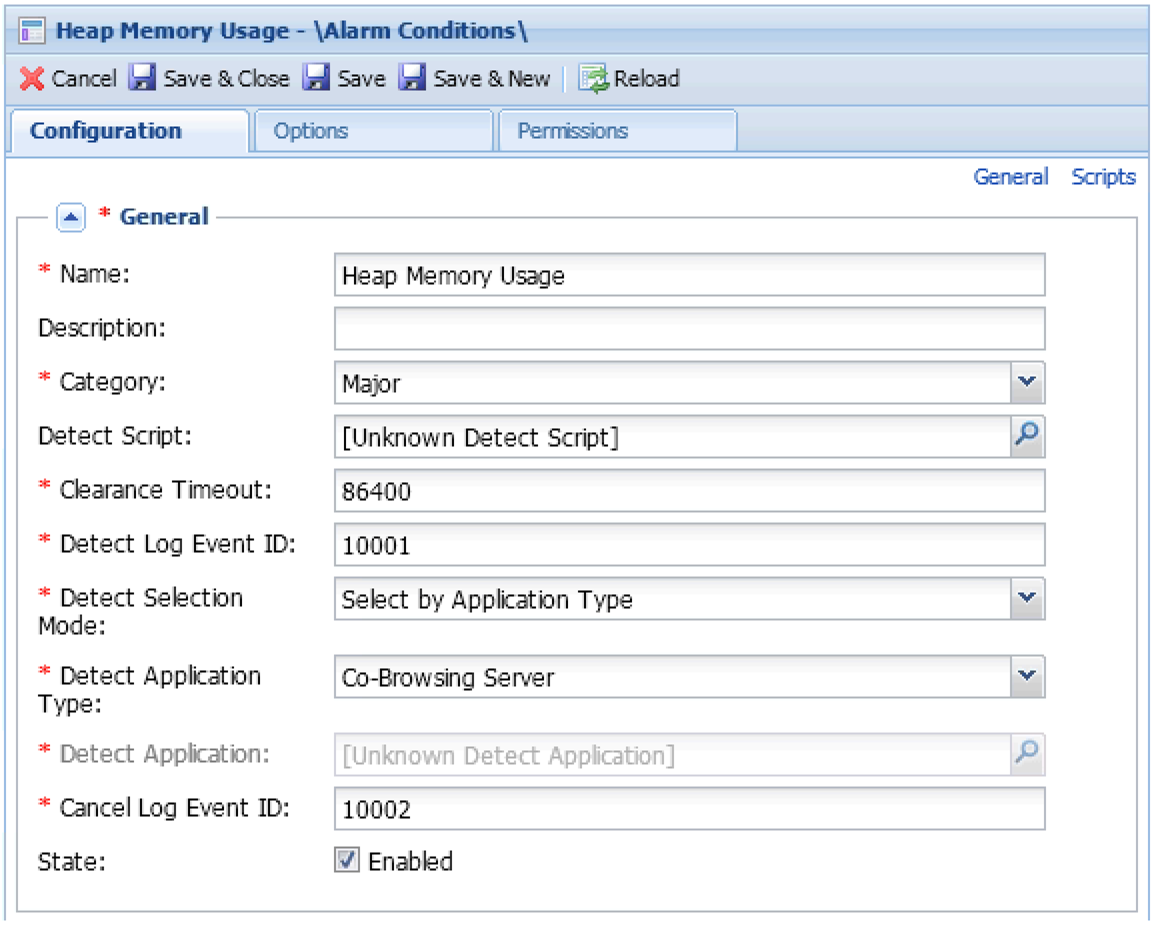
Configuring Alarm Condition Object

Message Server reporter needs each predefined threshold to have a related **Alarm Condition** object in the Genesys Configuration.

While each predefined alarm can contain dedicated Alarm Condition object, only one Alarm Condition object is allowed for generic alerts because their Detect Log Event ID is the same.

You must manually create Alarm Condition objects in the **Alarm Conditions** section of Genesys Administrator:

Configuring an Alarm Condition Object in Genesys Administrator



* Name:	Heap Memory Usage
Description:	
* Category:	Major
Detect Script:	[Unknown Detect Script]
* Clearance Timeout:	86400
* Detect Log Event ID:	10001
* Detect Selection Mode:	Select by Application Type
* Detect Application Type:	Co-Browsing Server
* Detect Application:	[Unknown Detect Application]
* Cancel Log Event ID:	10002
State:	✓ Enabled

1. Open the **Provisioning > Environment > Alarm Conditions** section in Genesys Administrator.
2. Click **New** to create a new object.
3. Specify a **Name**. The value can be any string.
4. Set the proper **Detect Log Event ID** and **CancelLog Event ID**, see the Co-browse alarms configuration table.
5. Set **Select by Application Type** to `Detect Selection Mode`.
6. Set `Co-Browsing Server` for **Detect Application Type**.
7. Save your changes.

Important

For *generic alarms*, you should leave the **Cancel Log Event ID** empty and set a smaller **Clearance Timeout** because generic alarms have no Cancel Log Event ID and they cannot be automatically deleted from the Active Alarms view.

Configuring the threshold option in the server configuration

Co-browse server configuration contains the following threshold options:

- Predefined
    - HeapMemoryUsage.threshold
    - GcFrequency.threshold
    - GcLatency.threshold
    - InactiveSessions.threshold
    - JettyThreadPoolUsage.threshold
    - ServerResponseTime.threshold
    - SlaveRenderLatency.threshold
- *Generic* threshold configurations use the option <metricName>.threshold.

To configure a predefined threshold set the proper value for the corresponding option.

To configure a generic threshold:

1. Substitute the metric name placeholder with the actual metric name, see Breakdown of Available Metrics.
2. Set the proper value for the metric's threshold.

Co-browse Alarms Configuration Table

Alarm name	**Alarm Condition object**					**Related configuration option, `metrics` section**		
Threshold type	**Selection mode**	**Application type**	**Detect Event ID**	**Cancel Event ID**	**Option**	**Default value**	**Description**	
Heap Memory Usage	predefined	Select by Application Type	Co-browse Server	10001	10002	HeapMemoryUsage.threshold	0.8	Defines heap memory usage threshold value. This is the ratio of the used heap memory to the maximum heap memory.

Alarm name	Alarm Condition object					Related configuration option, `metrics` section		
GC Frequency				10003	10004	GcFrequency.threshold	24	Defines GC frequency threshold value for an hour.
GC Latency				10005	10006	GcLatency.threshold	1000	Defines GC Latency threshold value, in milliseconds, in relation to the last GC occurred in the configured time interval.
Inactive Sessions				100001	100002	InactiveSessions.threshold	0.2	Defines the ratio of inactive sessions to all sessions from the configured time interval. It shows how many Co-browse sessions were created by customer but never joined by an agent.
Slave Render Latency				100003	100004	SlaveRenderLatency.threshold	10000	Defines, in milliseconds, the SlaveRenderLatency metric threshold

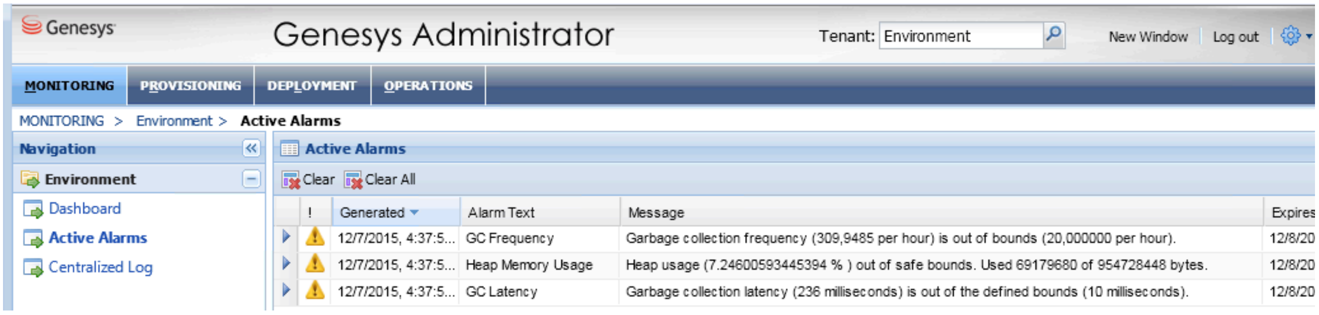
Alarm name	Alarm Condition object					Related configuration option, `metrics` section		
								value in the configured time interval. Agent side rendering latency shows whether reported agent side rendering is too slow.
Jetty Thread Pool Usage				100005	100006	JettyThreadPoolUsage.threshold	0.9	Defines Jetty thread pool usage threshold value. This is the ratio of the used Jetty thread pool size to the maximum available. It signals whether too few free threads handle http requests.
Server Response Time				100007	100008	ServerResponseTime.threshold	100	Defines, in milliseconds, the maximum value allowed for ServerResponseTime metric. The metric is

Alarm name	Alarm Condition object					Related configuration option, `metrics` section		
								calculated as average time for the latest N routings of data from customer to agent, where N is defined by the ServerResponseTime.slid option value.
						ServerResponseTime.slidingWindowSize	1000	Defines the number of recent measurements applied for the ServerResponseTime metric calculation.
Generic alarm	generic			10007		Generic threshold option		Defines threshold value for the particular metric.

Using Alarms to Improve Co-browse Performance

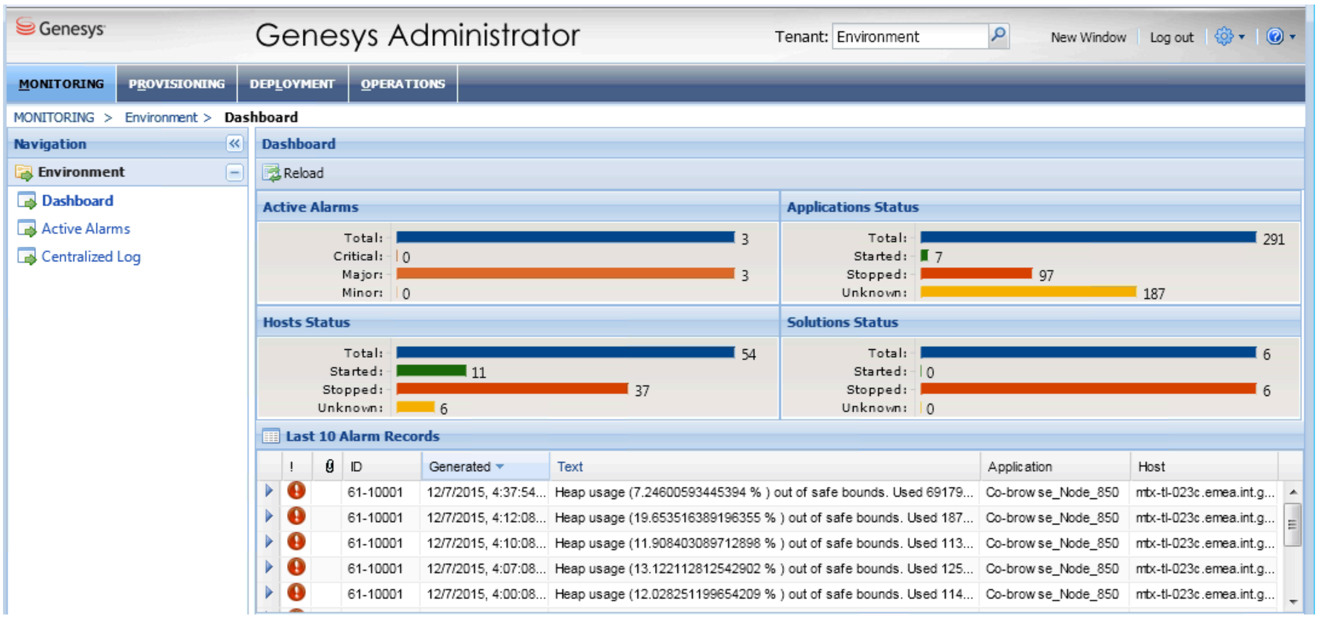
Co-browse Alarm Reporting

Once you configure your Co-browse alarm reporting, you can monitor your Co-browse Server:



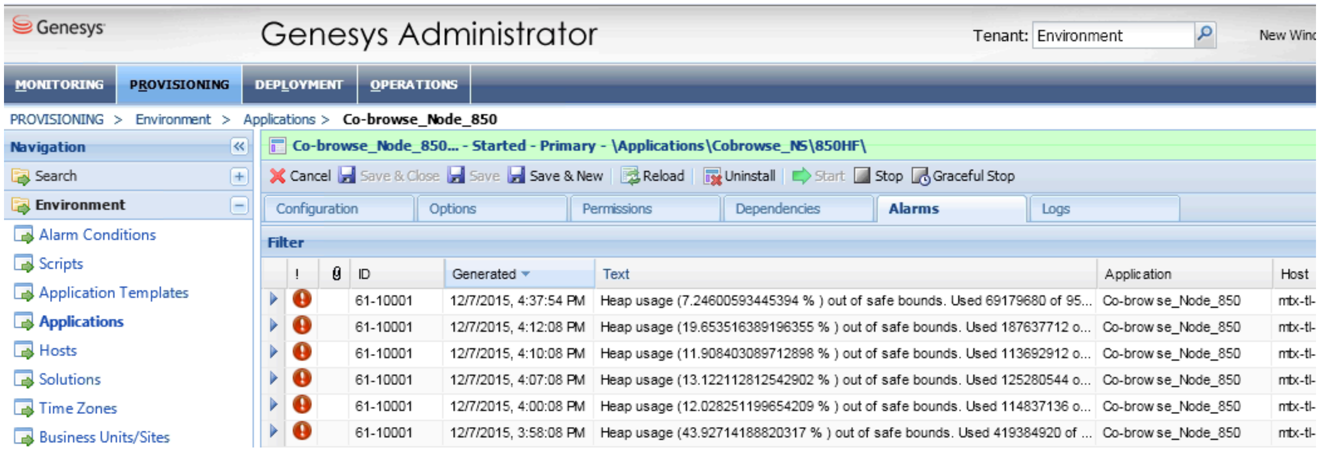
	!	Generated	Alarm Text	Message	Expires
▶	⚠	12/7/2015, 4:37:5...	GC Frequency	Garbage collection frequency (309,9485 per hour) is out of bounds (20,000000 per hour).	12/8/20
▶	⚠	12/7/2015, 4:37:5...	Heap Memory Usage	Heap usage (7.24600593445394 %) out of safe bounds. Used 69179680 of 954728448 bytes.	12/8/20
▶	⚠	12/7/2015, 4:37:5...	GC Latency	Garbage collection latency (236 milliseconds) is out of the defined bounds (10 milliseconds).	12/8/20

You can observe all monitoring alarms in the **Active Alarms** section of Genesys Administrator.



	!		ID	Generated	Text	Application	Host
▶	●		61-10001	12/7/2015, 4:37:54...	Heap usage (7.24600593445394 %) out of safe bounds. Used 69179...	Co-browse_Node_850	mtx-tl-023c.emea.int.g...
▶	●		61-10001	12/7/2015, 4:12:08...	Heap usage (19.653516389196355 %) out of safe bounds. Used 187...	Co-browse_Node_850	mtx-tl-023c.emea.int.g...
▶	●		61-10001	12/7/2015, 4:10:08...	Heap usage (11.908403089712898 %) out of safe bounds. Used 113...	Co-browse_Node_850	mtx-tl-023c.emea.int.g...
▶	●		61-10001	12/7/2015, 4:07:08...	Heap usage (13.122112812542902 %) out of safe bounds. Used 125...	Co-browse_Node_850	mtx-tl-023c.emea.int.g...
▶	●		61-10001	12/7/2015, 4:00:08...	Heap usage (12.028251199654209 %) out of safe bounds. Used 114...	Co-browse_Node_850	mtx-tl-023c.emea.int.g...

You can observe fatal alarms in the Genesys Administrator **Dashboard**.



!		ID	Generated	Text	Application	Host
●		61-10001	12/7/2015, 4:37:54 PM	Heap usage (7.24600593445394 %) out of safe bounds. Used 69179680 of 95...	Co-brow se_Node_850	mtx-tl-
●		61-10001	12/7/2015, 4:12:08 PM	Heap usage (19.653516389196355 %) out of safe bounds. Used 187637712 o...	Co-brow se_Node_850	mtx-tl-
●		61-10001	12/7/2015, 4:10:08 PM	Heap usage (11.908403089712898 %) out of safe bounds. Used 113692912 o...	Co-brow se_Node_850	mtx-tl-
●		61-10001	12/7/2015, 4:07:08 PM	Heap usage (13.122112812542902 %) out of safe bounds. Used 125280544 o...	Co-brow se_Node_850	mtx-tl-
●		61-10001	12/7/2015, 4:00:08 PM	Heap usage (12.028251199654209 %) out of safe bounds. Used 114837136 o...	Co-brow se_Node_850	mtx-tl-
●		61-10001	12/7/2015, 3:58:08 PM	Heap usage (43.92714188820317 %) out of safe bounds. Used 419384920 of ...	Co-brow se_Node_850	mtx-tl-

You can also observe fatal alarms in the **Alarms** tab of your *Co-browse node*'s application properties.

Responding to Co-browse Alarms

Monitoring alarms detect problems in your application server. The table below lists possible actions to resolve problems detected.

Once you fix a problem, the server recalculates the metric after the monitoring time interval and deletes the alarm from the alarm monitoring view. At the same time, the appropriate message appears in the log and states that the metric value is back to normal.

Actions to Respond to Common Alarms

Alarm name	Fatal?	Detect alarm message example	Problem description	Actions to fix the problem	Cancel alarm message
Heap Memory Usage	yes	[ERROR] HeapUsageThreshold - Heap usage (40.65 %) out of safe bounds. Used 388140568 of 954728448 bytes.	This alarm signals that Co-browse Server is working but at full capacity.	To prevent the application from overloading, you should extent the memory heap: 1. Open setenv.bat (Windows) or setenv.sh (UNIX) for editing. 2. Increase **Xmx*** value in the **JAVA_OPTS** directive: `set JAVA_OPTS=%JAVA_OPTS`	[WARN] HeapUsageThreshold - Heap usage (30.05 %) is back to normal

Alarm name	Fatal?	Detect alarm message example	Problem description	Actions to fix the problem	Cancel alarm message

				`% ...`
				`-Xmx1024m`
				`...`
				3. Restart the Co-browse Server application.	

GC Frequency	no	[ERROR] GcFrequencyThreshold - Garbage collection frequency (24,4718 per hour) is out of bounds (24,000000 per hour).	There might be several causes: • The heap memory size is less than needed • Too many created entities. It might happen due to log message overloading • If this problem happened while the log level is high, the reason might be hyperactivity of sessions while the memory heap is small.	1. You should increase heap size as described above. 2. Increasing the log level may resolve the problem. If these solutions do not help, you should add key **Xmn*** in the **JAVA_OPTS** directive in setenv.bat/sh file.	[WARN] GcFrequencyThreshold - Garbage collection frequency (20.6773 per hour) is back to normal
GC Latency	no	[ERROR] GcLatencyThreshold - Garbage collection latency (<number> milliseconds) is out of the defined bounds (<number> milliseconds).	This alarm means that the GC processor is overloaded.	To resolve the problem, you should remove excessive load by either: • replacing existent processor with a more	[WARN] GcLatencyThreshold - Garbage collection latency (251 milliseconds) is back to normal

Alarm name	Fatal?	Detect alarm message example	Problem description	Actions to fix the problem	Cancel alarm message
				powerful one • replacing existent RAM with faster RAM • doing both.	
Inactive Sessions	no	[ERROR] InactiveSessionsThreshold - Percent of inactive sessions 0,25 out of bounds 0,20. 10 are inactive from 40	Shows how many Co-browse sessions were created but never joined by an agent.		[WARN] InactiveSessionsThreshold - Percent of inactive sessions is back to normal
Slave Render Latency	no	[ERROR] SlaveRenderLatencyThreshold - Average time of slave rendering (14730,0 milliseconds) is out of bounds (10000 milliseconds)	This alarm alerts that the reported agent side rendering is too slow.		[WARN] SlaveRenderLatencyThreshold - Average time of slave rendering is back to normal
Jetty Thread Pool Usage	no	[ERROR] JettyThreadPoolUsageThreshold - Jetty thread pool usage (0,06) is out of bounds (0,001) . 11 busy threads from 200	Too few free threads allowed to handle http requests.		[WARN] JettyThreadPoolUsageThreshol - Jetty thread pool usage is back to normal
Server Responce Time	no	[ERROR] ServerResponseTimeThreshold - Average response time (68,68280 milliseconds) is out o f bounds (0,50000 milliseconds)	Co-browse Server may have exceeded the threshold because: • Co-browse Server is overloaded • The disk is working very slow	1. See the solution for the Heap Memory Usage alarm 2. Replace the disk with a more powerful one	[WARN] ServerResponseTimeThreshol - Average response time is back to normal

Important
To properly use the **Xmx**, **Xmn**, and **Xms** java options consult the Oracle documentation.

Localizing Co-browse Alarms

You can localize alarm log messages using LMS files. You can have two types of LMS files, an LMS file that includes common log messages and a project specific LMS file. Default LMS files are embedded into the Co-browse Server code.

To change the log message text, use the custom LMS files shipped with the product in the **<Co-browse Server root>/server/launcher** directory:

- **GeneralAlarms_en.lms** is a common LMS file
- **CobrowseAlarms_en.lms** contains project-specific log messages

To add localization to your monitoring alarms, apply the following to each custom LMS file:

1. Copy the content of the file to a new file name which ends with a system locale abbreviation. For example, au for Australia and `fr` for France. The common LMS file name for Australia would be `GeneralAlarms_au.lms`.
2. Edit the new file to change the log message text. Save your changes.
3. After you have finished editing each custom LMS file, restart the application server.

Important
To avoid inconsistency in alarm logging, the only thing you can change in a custom LMS file is the log message text.