Developer's Guide

Generating and Configuring the Instrumentation Script

5/11/2026

Generating and Configuring the Instrumentation Script

Contents

Overview

The Tracker Application instrumentation script is a small piece of JavaScript code that you paste into your website to enable Web Engagement functionality. You create this script by using the Script Generator in the Genesys Web Engagement Plug-in for Genesys Administrator Extension.

You typically add the instrumentation script to your site when you are ready to move your application to a production environment with a Web Engagement cluster or if you need to configure the script that is used by the GWM Proxy. If you are working in a standalone deployment in a lab environment, you can use the default GWM Proxy implementation.

You can complete the steps on this page to do the following:

1. Generate the basic instrumentation script.
2. Configure the script, if necessary for your solution.
3. Add the script to your website.

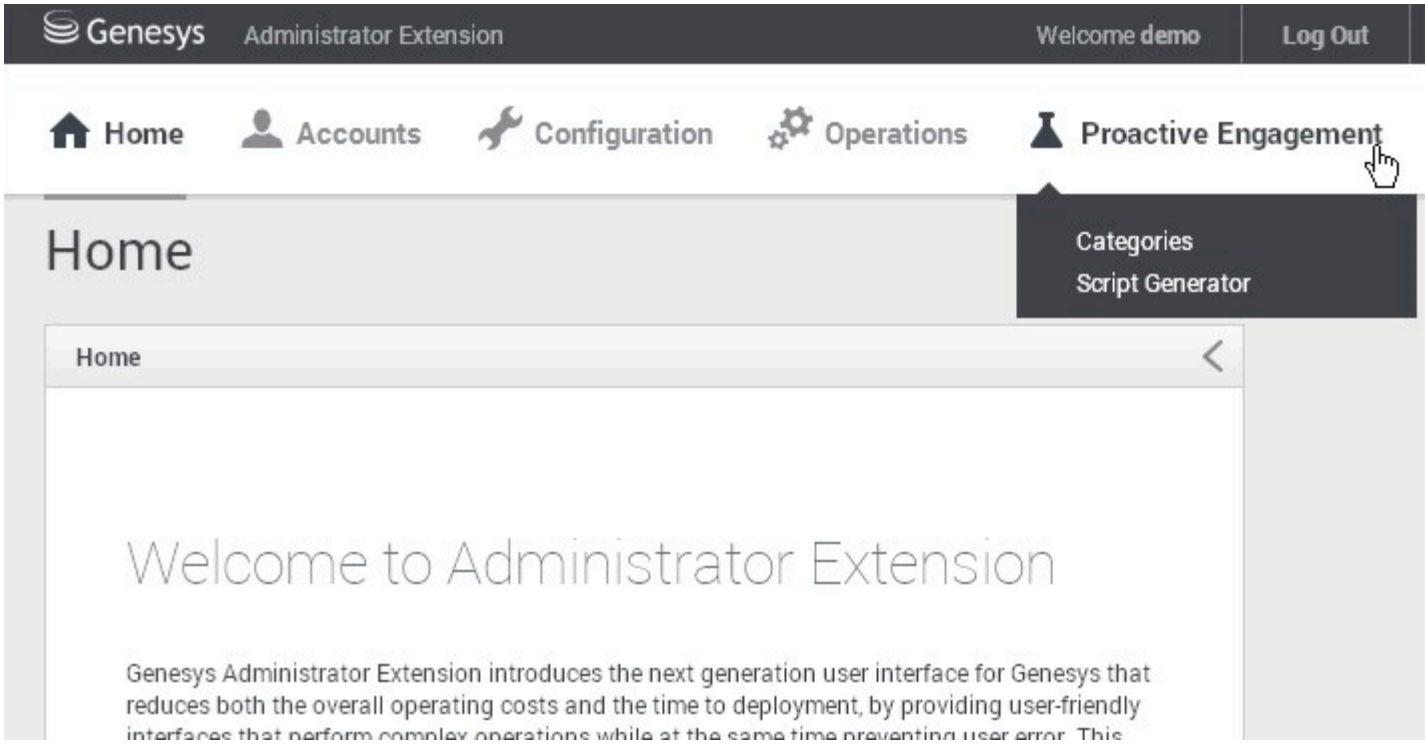
Generating the Instrumentation Script

Prerequisites

- You installed the Genesys Web Engagement Plug-in for Genesys Administrator Extension.

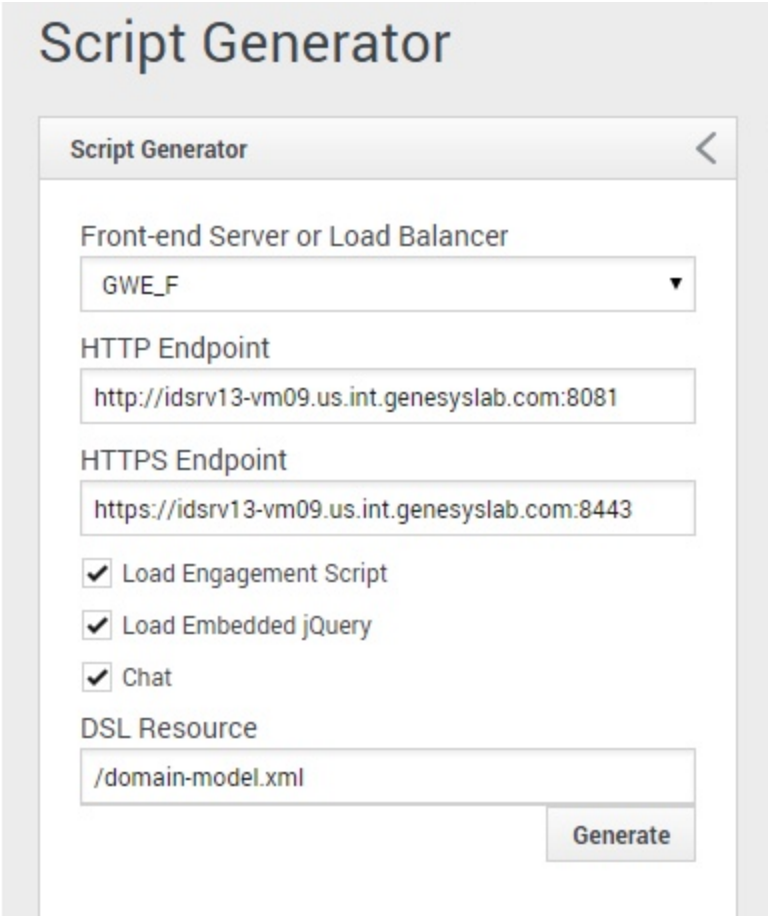
Start

1. Open Genesys Administrator Extension.



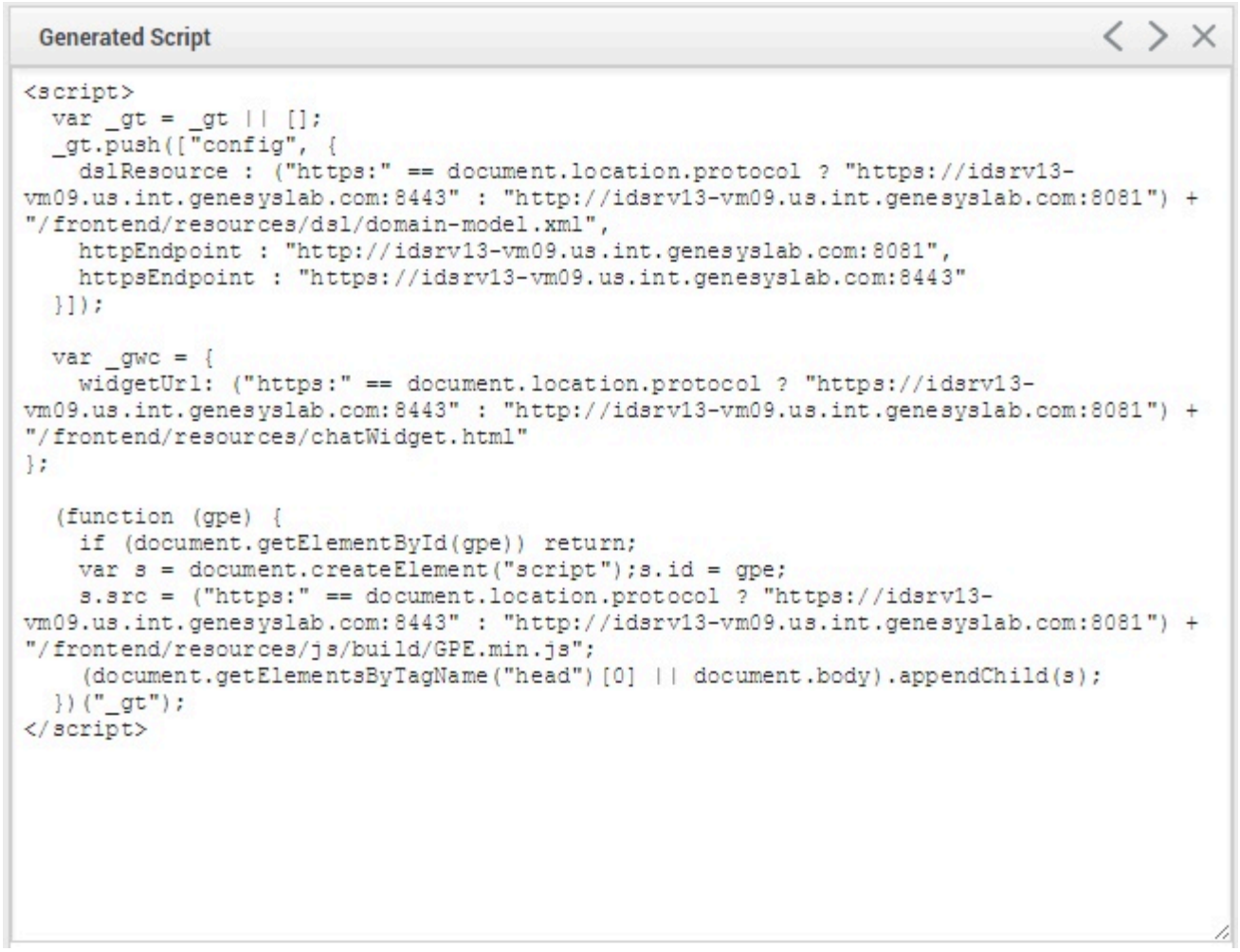
Main Home panel in the Genesys Administrator Extension

2. Navigate to **Proactive Engagement > Script Generator**. The **Script Generator** interface opens.
3. Fill in the following fields:
   Note: These values must be identical to the parameters you used to create your application. See Defining the Application's Monitoring Domains for details.
   - Select the correct Frontend Server or Load Balancer.
   - Enter the URL of the Frontend Server for the **HTTP Endpoint**. For example, `http://198.51.100.12:8081`
   - Enter the secure URL of the Frontend Server for the **HTTPS Endpoint**. For example, `http://198.51.100.12:8443`
   - Select **Load Engagement Script**, **Load Embedded jQuery**, and **Chat** to enable these features.
   - Enter the path(s) to the **DSL Resource**. The path is relative to the **/frontend/resources/dsl** directory of your Web Engagement application. You can add your DSL resources to this directory or sub-directories.



Example fields in the Script Generator.

4. Click **Generate**. The Generated Script panel opens and you can now copy your script.



The generated instrumentation script

If you are planning to configure the script, you might want to save it to a file so you don't lose your changes.

End

Next Steps

- You can configure your generated script.
- You can add the script to your website.

Back to top

Configuring the Instrumentation Script

The Tracker Application activates the Monitoring and Notification functions in Genesys Web

Engagement by inserting the **GTCJ.min.js** package into the page. This package includes jQuery, the Monitoring Agent, and the Notification Agent. The Tracker Application actually provides several packages that contain different functions and libraries. You can use these packages to enable different Web Engagement functionality on your website (these are added to your script when you use the GAX plug-in).

The table below shows the packages, in minified form, that are included with the Tracker Application.

Script	jQuery	Monitoring Agent	Notification Agent	Chat
GT.min.js	no	yes	no	no
GTJ.min.js	yes	yes	no	no
GTC.min.js	no	yes	yes	no
GTCJ.min.js	yes	yes	yes	no
GPE.min.js	yes	yes	yes	yes

> **Important**
>
> You must not make any changes to the scripts listed in the table above; any modifications will not be supported by Genesys. Please refer to the Genesys Web Engagement API Reference for information about the supported APIs.

The Tracker Application instrumentation script consists of two parts: configuration and script loader.

Script Loader

To load the Tracker Application, you just need to include the JavaScript in your web pages. This asynchronously loads the application, which means that it won't block other elements on your web pages from loading.

One solution for loading the script could be:

```
(function(gpe) {
    if (document.getElementById(gpe)) return;
    var s = document.createElement('script'); s.id = gpe;
    s.src = ('https:' == document.location.protocol ? 'https://<Frontend Server>:<Secure
Frontend Server Port>':
        'http://<Frontend Server>:<Frontend Server Port>') + '/frontend/resources/js/build/
GTCJ.min.js';
    (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('_gt');
```

> **Important**
>
> The script above uses the default "_gt" (Genesys Tracker) as the configuration global variable.

For more information about best practices for loading the script, see Adding the Instrumentation Script to Your Website.

Configuration

By default, the Tracker Application script uses the "_gt" (Genesys Tracker) global variable (you can change this in the script loader — see Changing the Global Configuration Variable for details) that must be initialized before the script loader is actually added to the page.

The following configuration options are available in the script:

Parameter	Required	Type	Default Value	Description	Example value
httpEndpoint	yes (if "httpsEndpoint" is undefined)	string	-	The URL of the Frontend Server.	http://genesyslab.com:8081
httpsEndpoint	yes (if "httpEndpoint" is undefined)	String	-	The secure URL of the Frontend Server.	https://genesyslab.com:8443
dslResource	no	string	-	The DSL resource location. If dslResource is not defined, then the DSL is not loaded.	http://genesyslab.com:8081/frontend/resources/dsl/domain-model.xml
name	no	string	-	Name of the application. This option is a part of the cloud multi-tenant, multi-domain system. Currently not used.	genesyslab
domainName	no	string	Second-level domain (SLD).	Name of the domain where the cookie is stored.	For the domain sub.genesys.com, the second-level domain is genesys.com
languageCode	no	string	en-US	Localization tag for language and region. Used for categorization.	en-US
debug	no	boolean	false	Show Monitoring Agent debug information in the browser	true

Parameter	Required	Type	Default Value	Description	Example value
				console.	
debugComet	no	boolean	false	Show CometD debug information in the browser console.	true
preventIframeMonitoring	no	boolean	false	If preventIframeMonitoring is true, the Monitoring Agent does not generate system and business events if the agent is loaded in an iframe. See preventIframeMonitoring for details.	true
disableWebSockets	no	boolean	false	Disable websockets transport for the notification agent. By default, the Notification Agent uses websocket transport when it is possible. Make sure that your load balancers support websocket connections; otherwise, disable it — Disabling Websocket CometD Transport.	true

Basic Configuration

Basic configuration is the default Tracking functionality:

```
var _gt = window._gt || [];
_gt.push(['config', {
        dslResource:    ('https:' == document.location.protocol ? 'https://server:securePort'
:
                        'http://server:port') + '/frontend/resources/dsl/domain-model.xml',
```

```
        httpEndpoint: 'http://server:port',
        httpsEndpoint: 'https://server:securePort'
}]);

(function(gpe) {
        if (document.getElementById(gpe)) return;
        var s = document.createElement('script'); s.id = gpe;
        s.src = (   'https:' == document.location.protocol ? 'https://server:securePort' :
                    'http://server:port') + '/frontend/resources/js/build/GTCJ.min.js';
        (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('_gt');
```

This snippet represents the minimum configuration needed to track a page asynchronously. The `_gt` (Genesys Tracker) object is what makes the asynchronous syntax possible. It acts as a queue, which is a first-in, first-out data structure that collects API calls until Genesys Web Engagement is ready to execute them. To add something to the queue, you can use the `_gt.push` method. See the Monitoring JS API for more information.

Basic Configuration with the Chat JS Application

If you select "Chat" in the GAX plug-in, it adds chat functionality to the basic configuration by loading the Chat JS Application. Your script should now look something like this:

```
var _gt = window._gt || [];
_gt.push(['config', {
    dslResource:    ('https:' == document.location.protocol ? 'https://server:securePort' :
                    'http://server:port') + '/frontend/resources/dsl/domain-model.xml',
    httpEndpoint: 'http://server:port',
    httpsEndpoint: 'https://server:securePort'
    }]);

var _gwc = {
    widgetUrl: ('https:' == document.location.protocol ? 'https://server:securePort' :
                    'http://server:port') + '/frontend/resources/chatWidget.html'
};

(function(gpe) {
    if (document.getElementById(gpe)) return;
    var s = document.createElement('script'); s.id = gpe;
    s.src = (   'https:' == document.location.protocol ? 'https://server:securePort' :
                'http://server:port') + '/frontend/resources/js/build/GPE.min.js';
    (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('_gt');
```

Advanced Configuration

The snippet below shows the instrumentation script with extended configuration (refer to the configuration options table for details):

```
var _gt = _gt || [];
_gt.push(['config', {
    name:             'demo',
    domainName:       'localhost',
    languageCode:     'en-US',
    dslResource:    ('https:' == document.location.protocol ? 'https://server:securePort':
        'http://server:port') + '/frontend/resources/dsl/domain-model.xml',
    httpEndpoint: 'http://server:port',
    httpsEndpoint: 'https://server:securePort'
    languageCode:     'en-US',
    debug:            true,
```

```
    debugCometd:       true,
    preventIframeMonitoring: true,
}]);

(function(gpe) {
    if (document.getElementById(gpe)) return;
    var s = document.createElement('script'); s.id = gpe;
    s.src = (   'https:' == document.location.protocol ? 'https://server:securePort' :
        'http://server:port') + '/frontend/resources/js/build/GTCJ.min.js';
    (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('_gt');
```

preventIframeMonitoring

Some websites have iframe (or frame) elements on the page. If a website is instrumented so that the Monitoring Agent is loaded on all web pages (even in an iframe), the agent generates events for all pages, including iframes. For example, this means that a page with an iframe generates two PageEntered events, one for the main page and one for the iframe.

To prevent this, you can use a special initialization parameter, `preventIframeMonitoring`. This parameter is optional and has a default value of `false`. If `true`, the Monitoring Agent does not generate system and business events if it is loaded in an iframe.

Changing the Global Configuration Variable

You can change the global configuration variable for the Tracker Application by using the `data-gpe-var` attribute. For example:

```
(function(gpe) {
    if (document.getElementById(gpe)) return;
    var s = document.createElement('script'); s.id = gpe;
    s.src = ('https:' == document.location.protocol ? 'https://server:securePort':
        'http://server:port') + '/frontend/resources/js/build/GTCJ.min.js';
    s.setAttribute('data-gpe-var', gpe); // set global variable name for Tracker Application
    (document.getElementsByTagName('head')[0] || document.body).appendChild(s);
})('_myVariable');
```

In the example above global variable "_myVariable" is now used instead of "_gt".

Providing an External jQuery Library

If you already have a jQuery library on your website, you can reduce the size of the Genesys Web Engagement JavaScript files by using the packages without jQuery (**GT.min.js** or **GTC.min.js**). In this case, make sure that jQuery is available on your site through the global variable `window.jQuery` and that jQuery is loaded before the Genesys Tracker Application.

If the jQuery library is present on some pages and not others, you must insert the following snippet of code before the instrumentation script:

```
<script>
window.jQuery || document.write("<script src='http://code.jquery.com/
jquery-1.11.0.min.js'>\x3C/script>")
</script>
```

Disabling Websocket CometD Transport

To disable websockets CometD transport, use the **disableWebSockets** option in your instrumentation script:

```
_gt.push(['config', {
    disableWebSockets: true,
    ....
}]);
```

Enable A New Trigger Combined With A Previous Trigger

DSL is a great tool for creating business events on your website without the need for programming skills. But there are some use cases where you really need a JavaScript API. For these situations, you can use the Web Engagement Monitoring API.

The following example uses that API to set up a new trigger that is activated after another trigger has been activated. In this example, we are assuming that you have a web page that contains a text field and a submit button.

The first trigger is activated if a customer starts typing in the text field. If 100 seconds pass without the customer submitting their input, then I want to report that event to the Web Engagement server, so I need to set up a second trigger to create this action.

Here's one way to do it:

```
...
<p><input class="comment" type="text"></p>
<p><input class="submit" type="button" value="submit"></p>

<script>
    var timeout;

    $('.comment').focus(function() {
        if (!timeout) {
            console.log('timer started');
            timeout = setTimeout(function () {
                console.log('send event');
                _gt.push(['event', {eventName: 'myEvent'}])
            }, 100 * 1000)
        }
    });

    $('.submit').click(function() {
        if (timeout) {
            console.log('clean timeout');
            window.clearTimeout(timeout);
            timeout = undefined;
        }
    });
</script>
...
```

Next Steps

- When you are satisfied with your script configuration, you can move on to either Adding the Instrumentation Script to Your Website or Customizing an Application (if you configured the script so it

can be used with the GWM Proxy).

Back to top

Adding the Instrumentation Script to Your Website

To add the instrumentation script, you need to have access to the source code for your website. If you already have an older version of the instrumentation script on your site, make sure you remove it from each page before you add the new one. If you have customizations you want to add back to your pages after you add the new snippet, you can use a text or HTML editor to open and save a copy of each file.

The instrumentation script is loaded asynchronously. One of the main advantages of the asynchronous script is that you can position it at the top of the HTML document. This increases the likelihood that the tracking beacon will be sent before the user leaves the page. Genesys recommends placing the script at the bottom of the **<head>** section for best performance.

For the best performance across all browsers, Genesys recommends that you position other scripts in your site either before the instrumentation script in the **<head>** section or after both the instrumentation script and all page content (at the bottom of the HTML body).

Make sure that the document type is defined in the head of each of your web pages. If it is not defined, Genesys Web Engagement will not work on your website.

```
<!DOCTYPE html>
```

Prerequisites

- You removed any older versions of the instrumentation script from your site.
- You generated the instrumentation script.

Start

1. Select and copy the generated script from GAX or from your own file, if you configured the script.
2. Paste the script at the bottom of the **<head>** section of your web pages:
   - You can do this manually on each web page that you want to monitor.
   - You can do this in the header template of your website, if you have one.
3. If your website includes additional scripts, do one of the following to optimize performance:
   - Place your scripts above the instrumentation script in the **<head>** section.
   - Make sure your scripts are located after the webpage contents (at the bottom of the **body** section).

End

Next Steps

- After you have generated the script and added it to your website (or the GWM Proxy configuration), you are ready to Customize an Application.

Back to top