Developer's Guide

High-Level Architecture

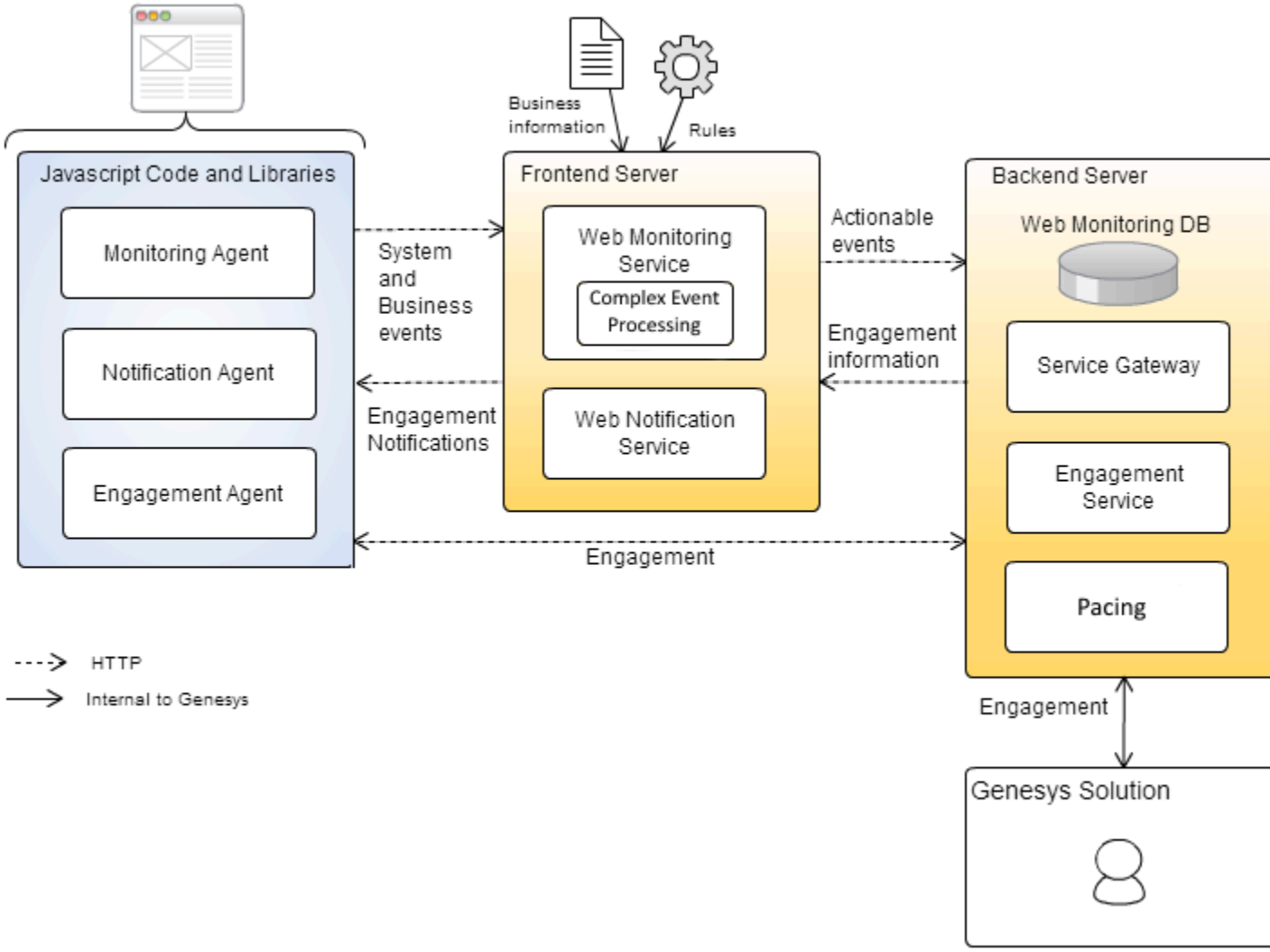
4/21/2025

High-Level Architecture

Contents

Genesys Web Engagement provides web services to interface your website with the Genesys contact center solution:

- The **Browser Tier widgets** and **Agents** are enabled by JavaScript code snippets that are inserted into your web pages; they run in the visitors' browser and track their browsing activity.
- The **Frontend Server** includes the Web Monitoring Service and the Web Notification Service, responsible for managing the data and event flow, based on a set of configurable rules and the visit's defined business events;
- The **Backend Server** stores data, submits information to the Genesys solution, and manages engagement requests to the Genesys contact center solution.



High-level architecture of Genesys Web Engagement

Browser Tier Agents

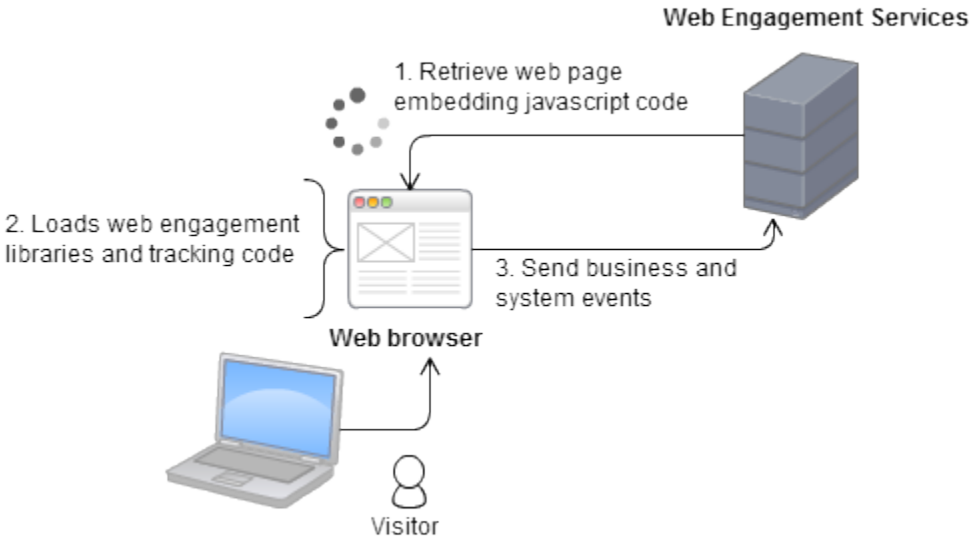
The Browser Tier Agents are implemented as JavaScript components that run in the visitor's browser. To enable monitoring on a web page, you create a short standardized section of JavaScript code with the Genesys Administrator Extension plug-in and then, you add this code to the <html></html> section of your web page.

When a customer visits the webpage, the code retrieved within the page loads all the necessary artifacts like the JavaScript libraries and Domain Specific Language (DSL) rules embedded in the JavaScript code.

The DSL rules cover:

- The HTML elements to monitor.
- The custom business events to send to the Frontend Server.
- The data to include in the events.

The Browser Tier generates categorized standard System and custom Business events, defined in the DSL definitions, and sends them to the Frontend Server over HTTP.



Browser interactions at runtime.

Genesys Web Engagement provides the following browser tier agents:

- The **Monitoring Agent** service records the web browsing activity. It generates basic system events such as VisitStarted, PageEntered, and additional custom business events, such as 'add-to-shopping cart'. These events are sent to the Web Engagement Frontend Server for further processing. For further information about events, see Event Workflow. For details about implementing monitoring, see Monitoring.
- The **Notification Agent** allows a web server to push data to a browser, without the browser explicitly requesting it, providing an asynchronous messaging channel between server and browser. It is used for presenting the engagement invite. For details about implementing notification, see Notification.

- The **Engagement Agent** provides the engagement mechanism, chat communication or web callback initialization. For details about implementing engagement, see Engagement.

If you are interested in monitoring features only, you do not need to install the Notification and Engagement Agent modules. Note that you cannot dynamically activate or deactivate the Notification and Engagement Agents on a visitor-by-visitor basis.

Web Engagement Frontend Server

The Genesys Web Engagement Frontend Server receives System and Business events from the browser's Monitoring Agent through its RESTful interface.

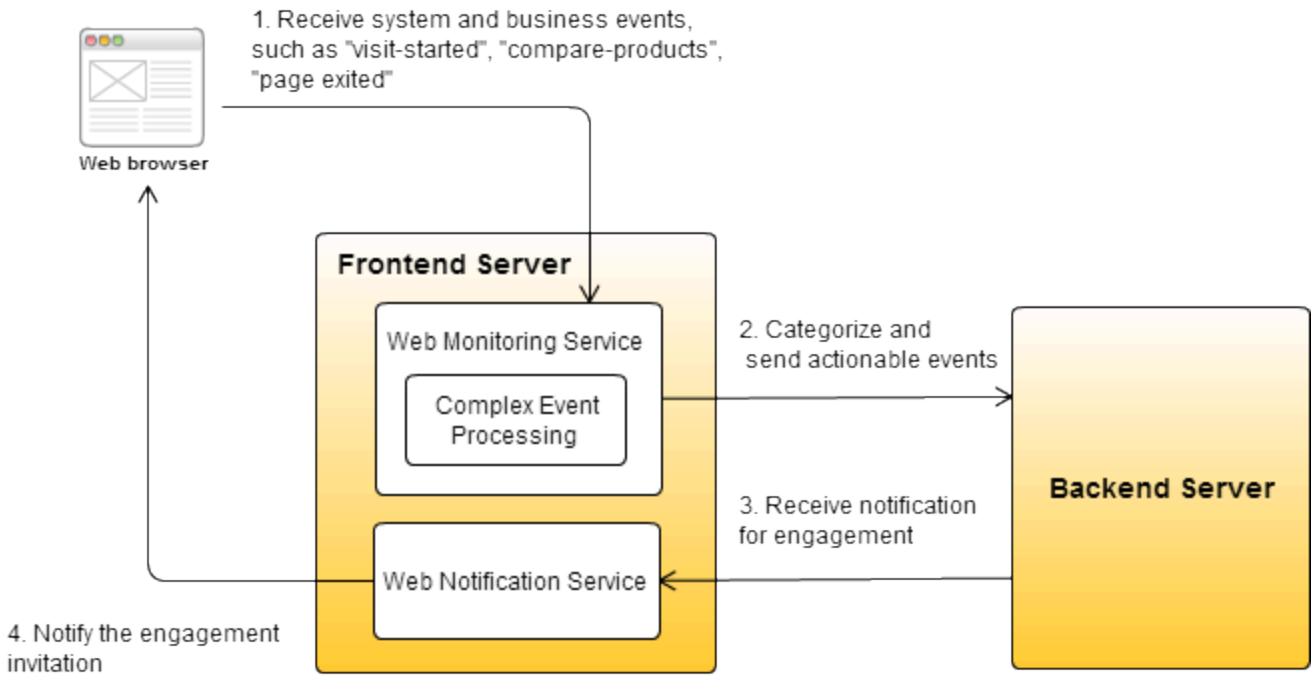
- **System** events are constants which cannot be customized. Two types of system events are available:
  - **Visit-related** events, such as VisitStarted or PageEntered;
  - **Identity-related** events, such as SignIn, SignOut, UserInfo.
    See Visitor Identification for further details.

Business events are additional custom events, that you can create by implementing Advanced Engagement:

- You can create and define them in the DSL loaded by the monitoring agents in the browser, with the Business Events DSL. For details about their implementation, read Managing Business Events.
- You can submit them from your web pages by using the Monitoring Javascript API.

For details about how the Business and System events are structured, see Events Structure.

The Frontend Server analyzes event correlation and attributes (such as the event name, event type, URL, or page title) and then assigns categories to the events. The integrated Complex Event Processing (CEP) engine validates these categorized events against the business rules and creates actionable events, which the Frontend Server sends to the Backend Server.



In addition, the Web Engagement Frontend Server also sends real-time invitation notifications to the Web Notification Agent of the web browser.

Hosting Static Resources

The Frontend Server is also responsible for hosting static resources, which are used in web applications such as: Invite Widget, Ads Widget, Chat Widget and so on. These resources are all available in the ***GWE_installation*\apps*application_name*_composer-project\WebEngagement_EngagementWidgets** folder.

```
-dsl
    -domain-model.xml   // - default DSL file
-js
    -chatAPI.min.js             // - chat service API
    -chatAPI-noTransport.min.js // - chat service API without transport
    -chatWidget.min.js          // - chat widget module
    -chatWidget-noDeps.min.js   // - chat widget module without dependencies
-locale
    -callback-en.json   // - default English localization file for callback widget
    -callback-fr.json   // - default Grench localization file for callback widget
ads.html                // - ads sample widget
callback.html           // - callback sample widget
chatTemplates.html      // - default chat template widget
chatWidget.html         // - chat widget file witch used in standalone chat mode
invite.html             // - invite sample widget
```

> **Important**

> You should not modify the **chatAPI.min.js**, **chatAPI-noTransport.min.js**, **chatWidget.min.js**, and **chatWidget-noDeps.min.js** files. They should be used if you build your solution using the Chat Service JS API or the Chat Widget JS API.

You can add your own static resources under the Frontend Server, but Genesys recommends you do this only if the resources are related to the Genesys Web Engagement solution. Alternatively, you can host your status resources under a third-party server, as long as it supports all the features required for the Web Engagement solution.

JSONP

The Frontend Server supports the JSONP protocol for all resources. JSONP stands for “JSON with Padding” and it is a workaround for loading data from different domains. It loads the script into the head of the DOM and thus you can access the information as if it were loaded on your own domain, by-passing the cross domain issue.

> **Tip**
> For more information about JSONP, see http://en.wikipedia.org/wiki/JSONP.

For example, for this request:

```
http://{fronend server}/frontend/resources/invite.html?obj=myObj&callback=myMethod
```

the server returns following response body:

```
myObj.myMethod('<content of http://{fronend server}/frontend/resources/invite.html>');
```

Cross-origin resource sharing

Cross-origin resource sharing (CORS) is a mechanism that allows many resources (for example, fonts, JavaScript, and so on) on a web page to be requested from another domain outside the domain from which the resource originated. In particular, JavaScript's AJAX calls can use the XMLHttpRequest mechanism. These "cross-domain" requests would otherwise be forbidden by web browsers due to the same-origin security policy.

> **Tip**
> For more information about cross-origin sharing, see http://en.wikipedia.org/wiki/Cross-origin_resource_sharing.

GZIP

The Frontend Server can serve pre-compressed static content as a transport encoding and avoid the

expense of on-the-fly compression. So if a request for **GPE.js** is received and the file **GPE.js.gz** exists, then it is served as **GPE.js** with a gzip transport encoding. By default, the Web Engagement solution ships all JavaScript resources in minified and pre-compressed version.

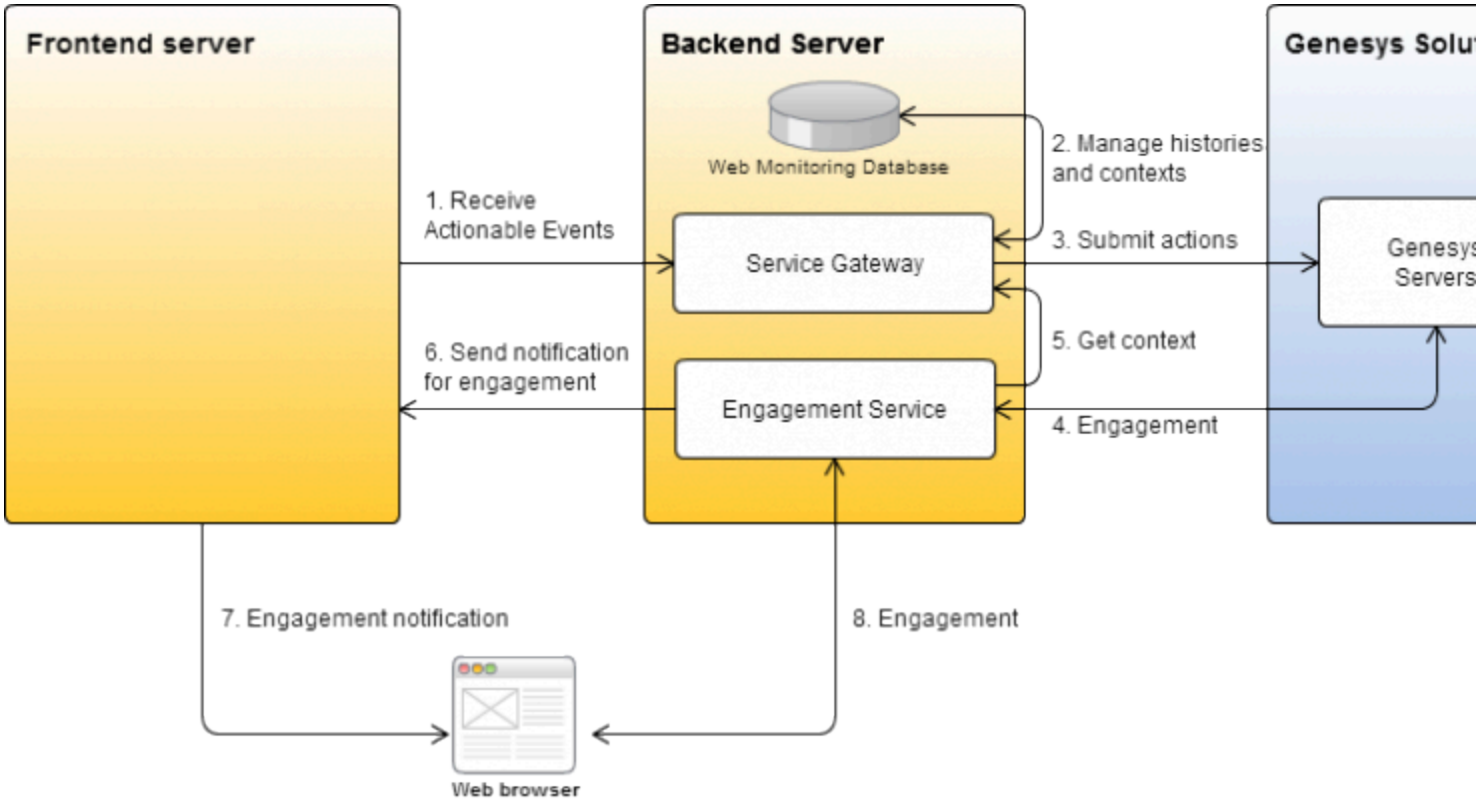
> ### Tip
> For more information about GZIP, see
>
> https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer#text-compression-with-gzip and
>
> http://en.wikipedia.org/wiki/HTTP_compression.

Web Engagement Backend Server

The Web Engagement Backend Server is the engagement's entry point to the Genesys servers. It delivers web and visitor information to the contact center, which allows that information to be correlated with contact information.

The Backend Server stores the events it receives from the Frontend Server, manages contexts and histories in its embedded Cassandra database, and submits them to the Genesys server.

When the Backend Server is notified that it should present a proactive offer, it retrieves the engagement information, based on the visit attributes. Then, if the SCXML strategies allow it, the proactive offer is displayed.

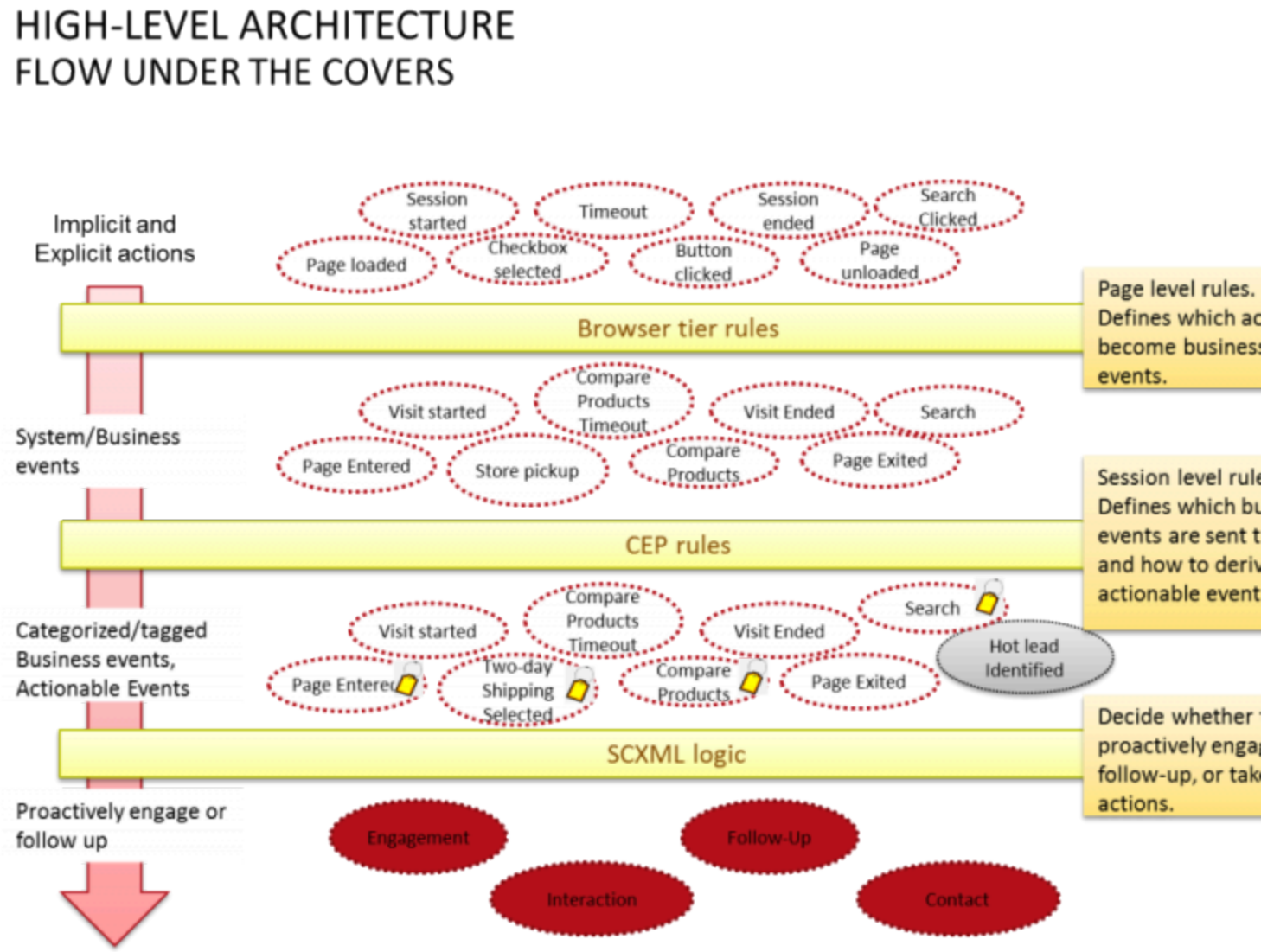


If the visitor accepts, the Engagement service connects to the Genesys servers. Once the connection is established, the service manages the engagement context information across the visit.

The Backend Server is also responsible for publishing rules for the Frontend Server. In a Standalone deployment you add a listening port for this purpose in Configuring Genesys Rules Authoring Tool. In a Clustering deployment scenario, you must configure one Backend Server to be in charge of rules deployment over the cluster - see Configuring Rules Deployment for the Cluster for details.

Event Workflow

The Genesys Web Engagement Frontend Server receives system and business events from the browser's Monitoring Agent. This event flow is used to create actionable events which generate requests to the Genesys solution, and make the engagement, follow up, and additional actions with the Genesys solution possible.



Event Flow Under the Covers

When a customer visits your website, he or she interacts with your web pages, which generates a flow of events, such as Session Started, Timeout, Button clicked, and so on.

The Monitoring Agent handles this traffic and translates all these events into the relevant System and Business events, according to your page rules, DSL, and category information.

The agent then submits the events to the Frontend Server where the Complex Event Processing embedded in the server determines the actionable events ("Hot lead Identified" in the above figure) and sends them to the Backend Server for further processing.

On the Backend Server, the SCXML strategies are used to determine whether to proactively engage, follow-up, or implement any other action.