Developer's Guide

High-Level Architecture

5/1/2025

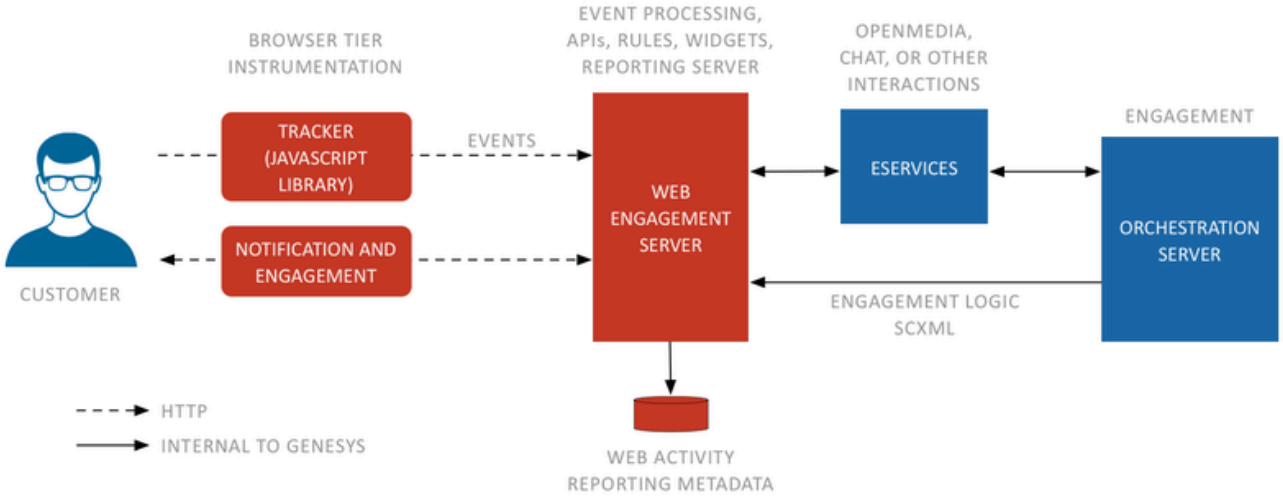
High-Level Architecture

Contents

Introduction

This article discusses the components that make up Genesys Web Engagement. Before you dive in, take a look at What is Web Engagement?

As mentioned in that article, Web Engagement has the following basic architecture:



As shown here, Web Engagement provides web services that connect your website with the Genesys contact center solution using:

- **Browser Tier Agents** (JavaScript code snippets) which are inserted into your web pages; they run in the visitor's browser and track their browsing activity.
- A **Web Engagement Server**, which includes the Web Monitoring Service and the Web Notification Service. This server is responsible for managing the data and event flow, based on a set of configurable rules and the visit's defined business events. It also stores data, submits information to the Genesys solution, and manages engagement requests to the Genesys contact center solution.

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, as specified in Configuring the Instrumentation Script, and then add this code snippet to the pages of your site.

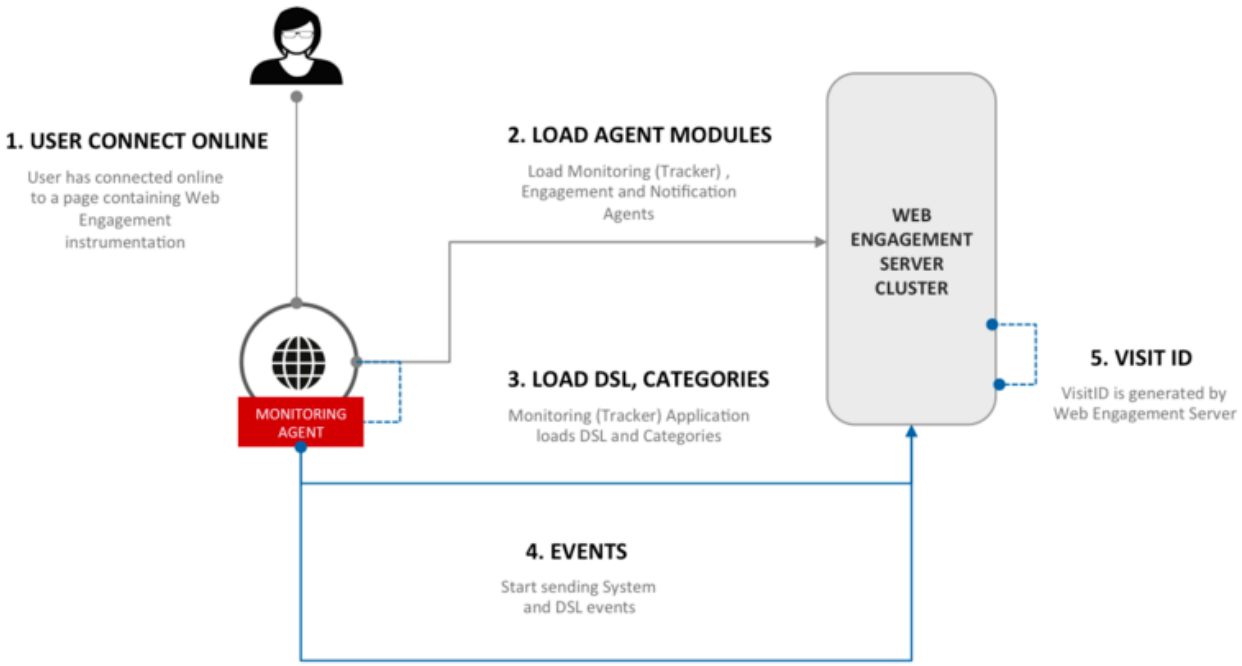
When a customer visits the webpage, the code retrieved within the page loads all the necessary artifacts like the JavaScript libraries and Domain Specific Language (DSL) that contains the definitions

of your Business Events.

The DSL covers:

- The HTML elements to monitor.
- The custom business events to send to the Web Engagement 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 Web Engagement Server over HTTP.



Genesys Web Engagement provides the following browser tier agents:

- The **Monitoring Agent** 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 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 and chat communication. For details about implementing engagement, see Engagement.

If you are only interested in Web Engagement's monitoring features, you need to configure your

instrumentation script accordingly. See Configuring the Instrumentation Script for details.

Web Engagement Server

Working with the Browser Tier

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

- **System** events track basic customer activities on your website. There are six of them, coming in two different flavors:
  - The **Visit-related** events, which are **VisitStarted**, **PageEntered**, and **PageExited**;
  - The **Identity-related** events, which are **SignIn**, **SignOut**, and **UserInfo**. See Visitor Identification for further details.
- **Business** events are additional custom events that you can create by implementing Advanced Engagement:
  - You create and define these events in the DSL loaded by the monitoring agents in the browser, using the Business Events DSL. For details about how to implement them, refer to Managing Business Events.
  - You can submit these events from your web pages by using the Monitoring Javascript API.

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

The Monitoring JavaScript Agent gets a list of categories from the Web Engagement Server and categorizes each event, based on the event data, prior to sending it to the server. The integrated Complex Event Processing (CEP) engine processes incoming events against the business rules and creates actionable events when the required conditions are met. For more information on rules, consult the documentation for Genesys Rules System.

The Web Engagement Server also sends invitation notifications to the Notification Agent injected into the visitor's browser.

Hosting Static Resources

The Web Engagement Server is also responsible for hosting static resources, which are used in web applications such as Invite Widget, Chat Widget, and so on. Just as it does in version 8.5, Web Engagement 8.5.1 has two types of resources: modifiable and non-modifiable. However, Web Engagement 8.5.1 handles different types of resources differently than the 8.5 version.

Non-modifiable static resources

Non-modifiable resources should not be changed by customers and always stay inside of the Web Engagement Server. They are available under the base path `http://{gwe server}/server/resources`

Non-modifiable resources include the Tracker JS Application, built-in version of Genesys CX Widgets,

and Chat JS Applications. Here is part of the resources structure inside of the Web Engagement Server:

```
-resources
    -css
        -widgets
            widgets.min.css
            widgets.min.css.gz
            ...
    -js
        -build
            GPE.min.js
            GPE.min.js.gz
            GT.min.js
            GT.min.js.gz
            ...
        -widgets
            cxbus.min.js
            widgets.min.js
            widgets.min.js.gz
            ...
        chatAPI-noDeps.min.js
        chatAPI-noDeps.min.js.gz
        ...
```

Examples:

Tracker JS application can be accessed with the following request: `http://{gwe server}/server/resources/js/build/GPE.min.js` Legacy Chat JS API can be accessed with the following request: `http://{gwe server}/server/resources/js/chatAPI-noDeps.min.js` Built-in CX Widgets can be accessed with the following request: `http://{gwe server}/server/resources/js/widgets/widgets.min.js`

Modifiable static resources

These resources are all available to the newly created Web Engagement application in the ***GWE_installation*\apps*application_name*\resources** folder. Customers can modify these resources and also add new resources to the structure. After deploying an active application into Web Engagement Server, these resources will be available under the base path `http://{gwe server}/server/api/resources/v1`

Note: When a new GWE application is deployed, all resources belonging to previously deployed applications are removed. **Note:** During the deployment process, modifiable resources will persist in the DB layer and are synchronized between all GWE nodes.

Here is sample of the modifiable resources structure created by Web Engagement Server for the newly created application:

```
-resources
    -_composer_projects      // GRDT and SCXML Composer projects
    -drl                     // Rule files specific for this application (for example,
playground application comes with "ready-to-use" DRL file and does not require GRAT
deployment procedfure)
    -dsl
        domain-model.xml     // default DSL file
    ads.html                 // sample advertisement widget
    chatTemplates.html       // scripts for template-based modification of chat widget
    chatWidget.html          // default chat widget
    invite.html              // default invitation widget
```

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

Examples:

Default DSL file can be accessed with this request: `http://{gwe server}/server/api/resources/v1/dsl/domain-model.xml` Default advertisement widget can be accessed with this request: `http://{gwe server}/server/api/resources/v1/ads.html`

JSONP

The Web Engagement 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://{gwe server}/server/api/resources/v1/invite.html?obj=myObj&callback=myMethod
```

the server returns following response body:

```
myObj.myMethod('<content of http://{gwe server}/server/api/resources/v1/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 Web Engagement 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.

Working with the Enterprise Tier

The Web Engagement Server is also 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.

On this end, the Web Engagement Server stores events, manages contexts and histories in its Cassandra database, and submits the appropriate data to the other Genesys servers.

When the Web Engagement 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 Web Engagement Server is also responsible for accepting rules deployed by the Genesys Rules Authoring Tool (GRAT).

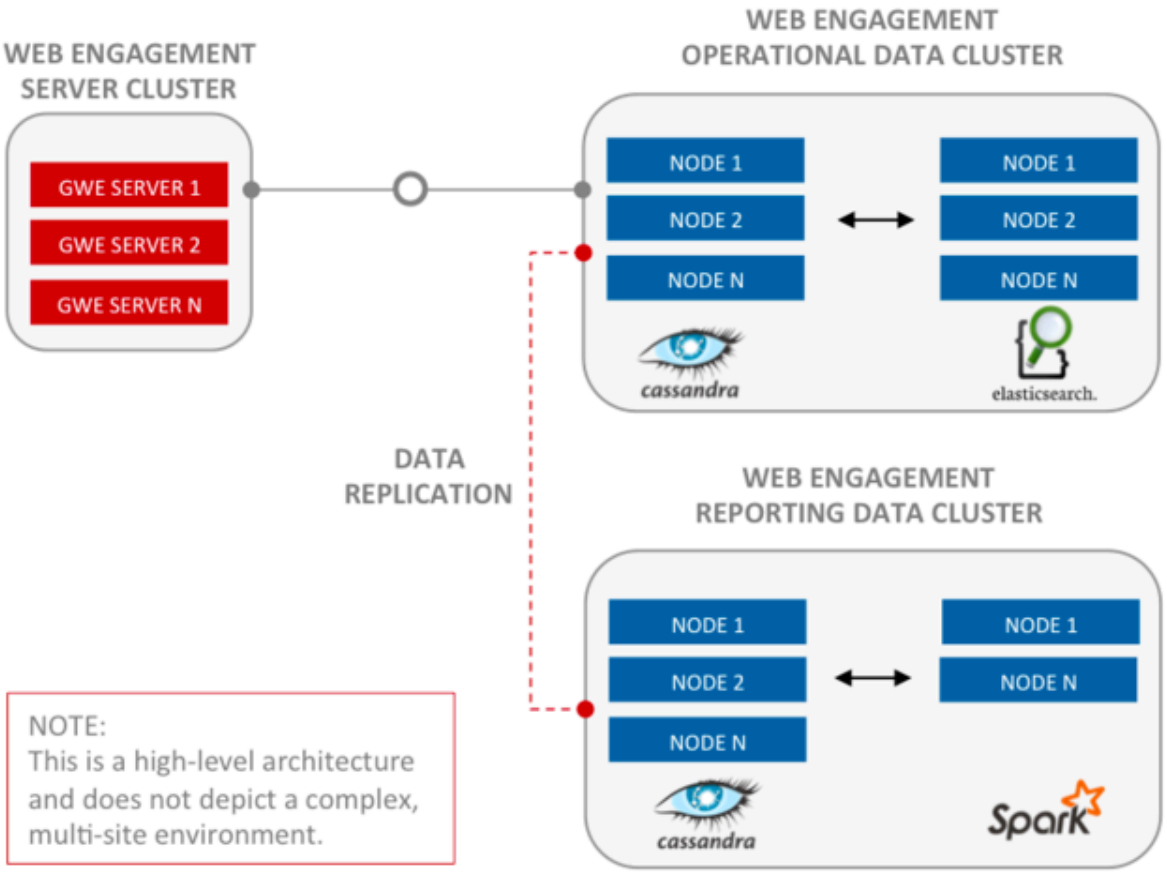
Database and Reporting

Web Engagement processes a large amount of data. To make this happen quickly enough, Genesys has combined three technologies into the database and reporting layers:

- Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.
- Elasticsearch is a search server that provides a distributed, multitenant-capable full-text search engine with a RESTful web interface and schema-free JSON documents.
- Apache Spark is an open source cluster computing framework.

Cassandra and Elasticsearch clusters are used in the Operational Cluster that stores data for realtime processing. This Cassandra data is indexed by Elasticsearch for quick access, and the combined results are replicated in a separate Cassandra cluster in the Reporting Cluster. This Reporting Cluster uses a Spark cluster that massages the data in the Cassandra reporting cluster for more sophisticated reporting.

The following diagram provides a highly simplified view of how it all fits together.



Event Workflow

The Genesys Web Engagement 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. (Note that an actionable event does not always result in a notification—sometimes an action could be "do nothing.")

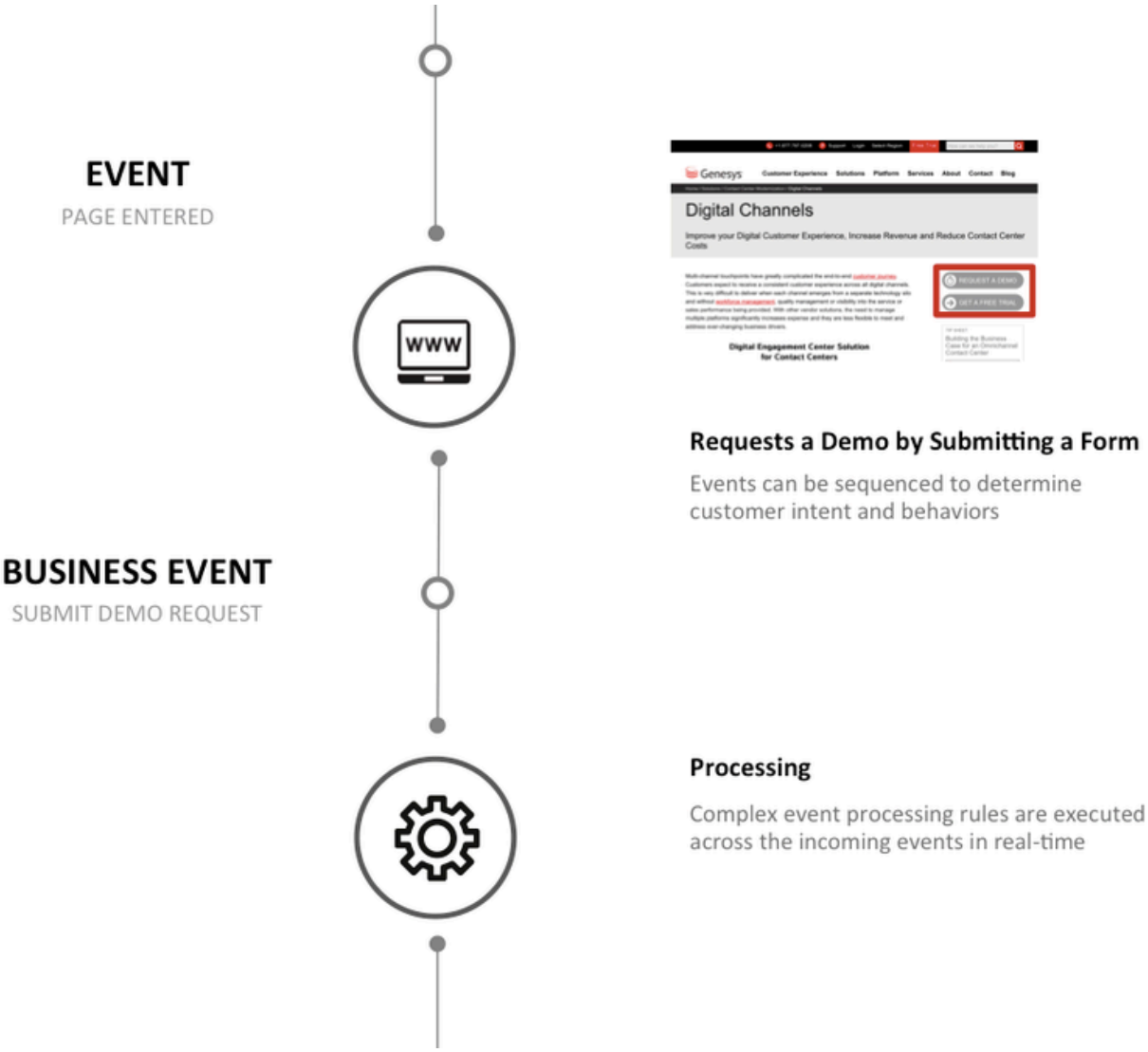
Here is a high-level view of this:

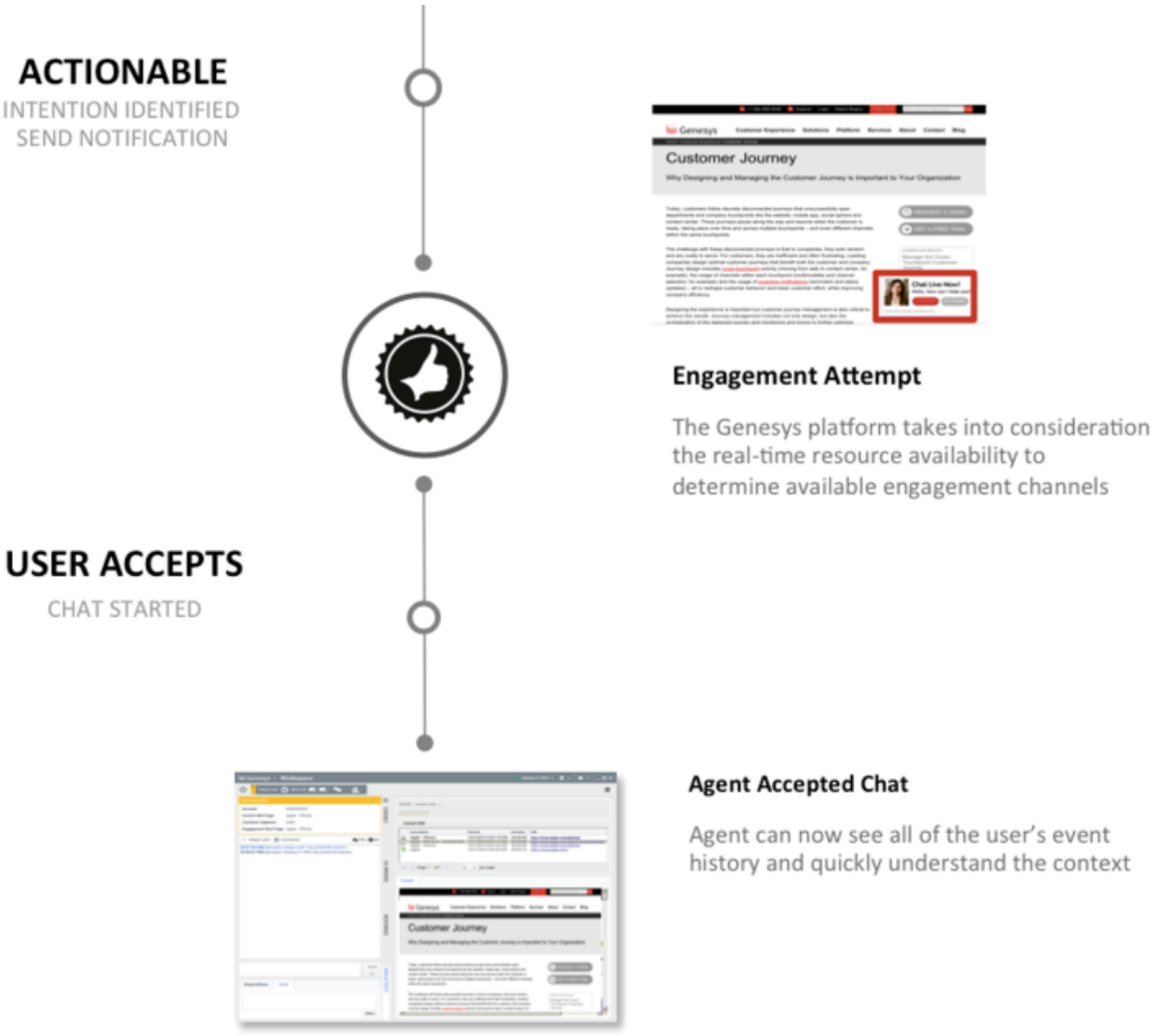
Connect Online

User has connected online; monitoring and engagement script is loaded

Browse to Product Page

As the user performs an activity within the app/ web/device, events are emitted to Genesys


EVENT
PAGE ENTERED
WWW
Genesys
Digital Channels
Improve your Digital Customer Experience, Increase Revenue and Reduce Contact Center Costs
Digital Engagement Center Solution for Contact Centers
Requests a Demo by Submitting a Form
Events can be sequenced to determine customer intent and behaviors
BUSINESS EVENT
SUBMIT DEMO REQUEST
Processing
Complex event processing rules are executed across the incoming events in real-time



As you can see, when a customer visits your website, he or she interacts with your web pages. The Monitoring Agent handles this traffic and translates it into the relevant System and Business events, according to your DSL and category information.

The agent then submits the events to the Web Engagement Server where the Complex Event Processing embedded in the server determines the actionable events ("Hot lead Identified" in the above figure) and carries out further processing. This includes the use of SCXML-based routing strategies to determine whether to proactively engage, to follow up, or to implement any other action.