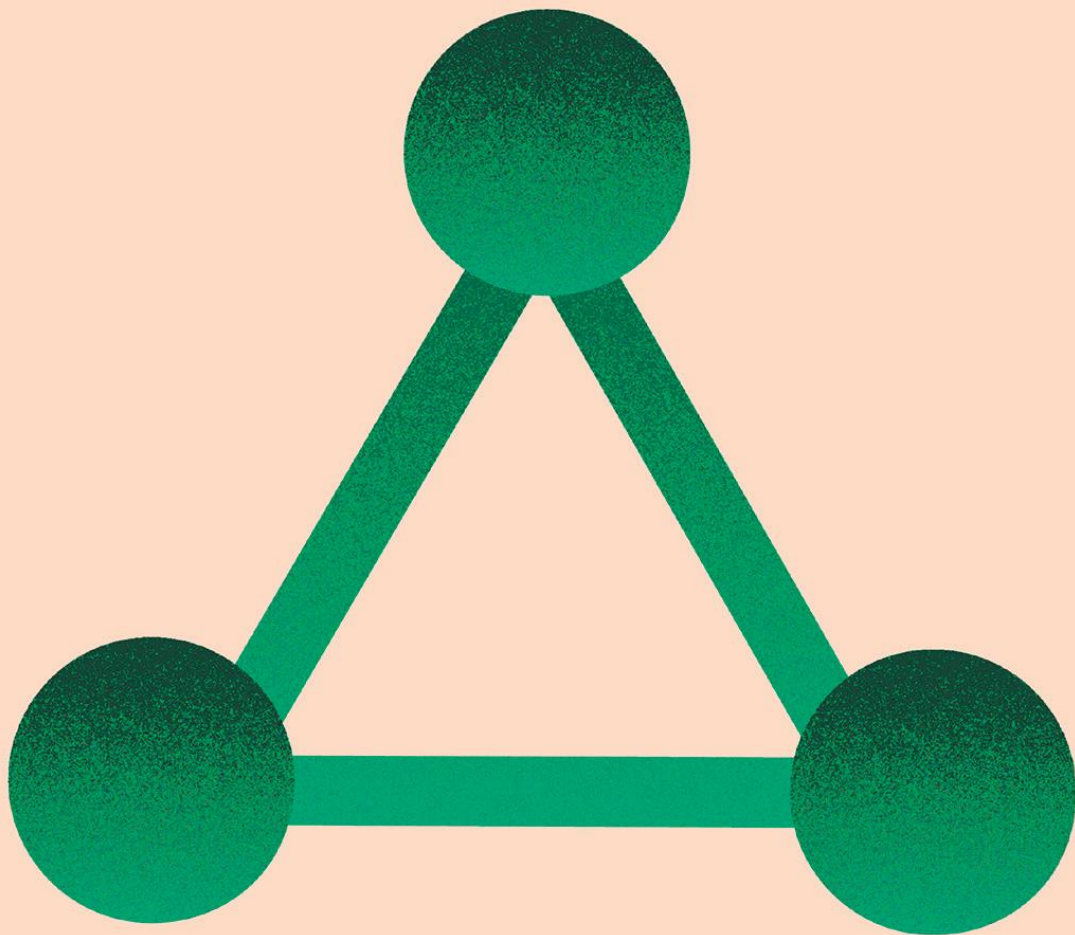


Queue•it

# Technical integration



Author: Martin Pronk  
 Creation Date: February 7, 2019  
 Last Updated: March 15, 2022  
 Version: 5.1.3

## Document Control

### Change Record

Date	Author	Reviewed by	Version	Change Reference
2019-02-07	Andrew Morris		4.3	Conversion to new template format
2019-01-04	Martin Pronk		4.3.1	Updated Good bots section.
2019-02-07	Johnny Packard		4.3.2	Updated Widgets: Target URL
2019-02-08	Johnny Packard		4.4	Added new Rejection Flow subsection
2019-09-20	Ismail Taouti		4.4.1	Updated section 4.2.1 with the latest user session cookie content. Updated section 9.2.3.2 Header Patterns. Updated section 11.4 FQDN regarding requesting SSL certificate for customized sub-domain. Updated section 6.5.1.3 with corrected code Updated section 11.3 with support for Hungarian and Serbian
2020-03-04	Skafti Jonsson		4.4.2	Updated section 4.1.3 Differentiate Between Regular and VIP Users
2020-06-03	Skafti Jonsson		4.4.3	Updated section 6.2 Added JavaScript Integration Snippet for Known User AJAX Protection. Added reference to Bots and Abuse Management White Paper and Proof-of-Work.
2020-10-05	Vasil Vasilev		4.4.4	Cookie domain not to be preceded by a . (dot) Updated SPA section, adding data-queueit-spa attribute
2020-12-18	Skafti Jonsson	Martin Larsen	5.0	Updated terminology. Sections 5.4, 10 and 11.3 updated
2021-04-09	Skafti Jonsson	Bruno Etchecopaz	5.1	Sections 5.3.4 and 10.1 updated Updated terminology. Added section 11.4 Multilingual Sites
			5.1.1	Section 4.6 updated
2022-02-03	Megan Henry	Svitlana Leus & Mojtaba Sarooghi	5.1.2	Added Section 5.5.2 Error Pages
2022-15-03	Svitlana Leus	Martin Larsen	5.1.3	Updated sections 4.1.3 and 5.5.2

## Table of Contents

1 Introduction .....	1
1.1 Scope .....	1
1.2 Prerequisites for this Document .....	1
1.3 Documentation Audience .....	1
1.4 Guide to this Document .....	1
1.4.1 Note on Terminology (as of March 2021) .....	2
1.5 Related Information .....	3
2 Protecting a Website .....	4
2.1 The User Journey .....	4
2.2 Controlling the Waiting Room .....	4
2.3 The Waiting room Life-Cycle .....	4
3 Enter Waiting Room Concept .....	6
3.1 Integration Configuration .....	6
3.1.1 Actions, Triggers and Configuration .....	6
3.1.2 Publish .....	6
3.2 Different Integration Methods .....	6
3.2.1 Known User .....	7
3.2.2 Queue-it JavaScript .....	7
3.2.3 Link To Queue .....	7
3.3 Combining Two Integration Methods .....	7
3.3.1 JavaScript and Link to Waiting Rooms.....	8
3.3.2 Known User and Link to Queue .....	8
4 Integration Configuration Details.....	9
4.1 Defining the right number of waiting rooms .....	9
4.1.1 Use case 1 – Localized Campaigns .....	9
4.1.2 Use case 2 – One waiting room with multiple brands / languages.....	10
4.1.3 Use case 3 – Differentiate between regular and VIP Visitors .....	10
4.1.4 Use case 4 – Protecting Specific Waiting Room Pages.....	11
4.1.5 Planning for Events / Campaigns / Flash Sales.....	12
4.2 Controlling the Visitor Session.....	13

4.2.1 Session Cookie Format and Content .....	14
4.2.2 User Journey Crosses Multiple Sub Domains .....	14
4.2.3 Canceling the Visitor Session .....	14
4.3 Setting Up Multiple Actions.....	15
4.3.1 The Order of Actions Matters .....	15
4.3.2 The “All Pages” Trigger .....	15
4.4 Handling e-mail / Facebook / Twitter Campaigns (Link to Waiting Room).....	15
4.5 Handling Call-backs from Payment Provider .....	16
4.6 Trigger Types.....	16
4.7 Action Types .....	17
4.7.1 Queue Action.....	17
4.7.2 Cancel Action.....	18
4.7.3 Ignore Action .....	18
4.8 Configurations .....	18
4.8.1 Special for the JavaScript Integration .....	19
5 Known User Integration Method Details.....	20
5.1.1 Known User Flow .....	20
5.1.2 Supported Languages.....	21
5.1.3 Configuring the Waiting Room(s).....	22
5.2 Know User Code flow .....	22
5.2.1 Read the waiting room configuration .....	22
5.2.2 Validate the <i>queueittoken</i> and Store a Session Cookie.....	23
5.2.3 Redirect the Visitor.....	23
5.2.4 Rejection flow .....	23
5.3 Known User Implementation.....	24
5.3.1 Accessing Configuration Via the GO Queue-it Platform .....	24
5.3.2 Accessing the Configuration as an Object in Code .....	26
5.3.3 pureUrl.....	28
5.3.4 Queue Configuration Downloader .....	28
5.3.5 Query String Parameters (Queue-it Token) .....	29
5.4 Known User Integration Test .....	31
5.4.1 Prerequisites .....	31
5.4.2 Successful Flow .....	32

5.4.3 No Looping Around .....	33
5.4.4 Tampered Link .....	34
5.4.5 Shared Link.....	35
5.5 Troubleshooting .....	35
5.5.1 Debug link.....	35
5.5.2 Error Pages .....	37
6 JavaScript Integration Method Details .....	40
6.1 JavaScript Integration .....	40
6.2 JavaScript Integration Snippets .....	40
6.2.1 Simple JavaScript Integration Snippet: .....	41
6.2.2 JavaScript Integration Snippet for Known User AJAX Protection.....	41
6.3 Benefits.....	41
6.3.1 Note: Queue-it Backend Response.....	42
6.4 Configuring Waiting Rooms .....	42
6.4.1 On the JavaScript Tab .....	42
6.5 Making Bypassing the Queue More Difficult.....	43
6.5.1 Method 1: When Using JQuery.....	43
6.5.2 Method 2: When Not Using JQuery .....	45
6.6 Queue-it Session Cookie .....	47
6.7 JavaScript Integration for Single Page App (SPA) .....	47
6.7.1 Integration for Single Page Applications that use URL navigation .....	48
7 Link To Waiting Room Integration Method Details.....	50
7.1 Parameters.....	50
8 Queue Skipping .....	52
8.1 Skip by IP Addresses Functionality.....	52
8.2 IP Address Groups.....	52
8.3 Bypass Link.....	52
8.4 Logging .....	52
9 Bot Management.....	54
9.1 Bot Categories and Types .....	54
9.1.1 Bot Types (Good vs. Bad Bots).....	54
9.1.2 Good Bots .....	54

9.1.3 Bad Bots.....	55
9.2 Bot Support in Queue-it .....	55
9.2.1 Robots.txt .....	55
9.2.2 Request Analysis .....	56
9.2.3 Let your good bots bypass Queue-it .....	57
9.2.4 Proof-of-Work Challenge .....	59
10 Visibility Modes.....	60
10.1 Visible At Peak.....	60
10.1.1 Optimization Options for Visible At Peak Mode.....	60
10.2 Always Visible.....	61
11 Custom Theme .....	62
11.1 Default Theme .....	62
11.2 Customized Theme.....	62
11.3 Language and Culture .....	62
11.4 Multilingual Sites .....	62
11.5 FQDN - (Using another FQDN for branding purpose) .....	63
12 Widgets.....	64
12.1 Enter Queue Button .....	64
12.1.1 Text .....	64
12.1.2 Styling.....	64
12.1.3 Target URL .....	65
12.1.4 Target URL Parameter .....	65
12.2 Countdown.....	65
12.2.1 Text .....	66
12.2.2 Styling.....	66
13 Queue Outflow .....	67
13.1 The Actual Redirect.....	67
13.2 Defining Capacity and the Flow-based Approach.....	69
13.2.1 Traditional Way of Defining Capacity .....	69
13.2.2 The Flow-based Approach.....	69
13.2.3 Set Release Time .....	70
13.3 The Estimated Wait Time Calculation .....	71

13.4 Understanding Outflow in Different Scenarios.....	72
13.4.1 Opening Scenario.....	72
13.4.2 Steady Flow .....	72
13.4.3 Steady Flow at Very Low Ratio .....	73
13.4.4 Sudden Drop in Short Period.....	74
13.4.5 Increasing Max Outflow .....	74
13.4.6 Decrease Max Outflow .....	75
14 Queue-it Availability .....	76

# 1 Introduction

The following sections describe the content and organization of the document:

- [Scope](#)
- [Prerequisites for this Document](#)
- [Documentation Audience](#)
- [Guide to this Document](#)
- [Related Information](#)

## 1.1 Scope

This document describes how the waiting room works and provides conceptual information that is helpful to understanding the necessary components.

## 1.2 Prerequisites for this Document

Prior to reading this white paper, readers should have a general understanding of the waiting room features. To become familiar with how the waiting room functions, participate in a [product demo](#) or contact [support@queue-it.com](mailto:support@queue-it.com).

## 1.3 Documentation Audience

This document is intended for the following audiences:

- Technical—Developers and other technical users with a focus on architecting, developing, and maintaining applications.

## 1.4 Guide to this Document

This document summarizes features and provides an overview of the architecture. This document is organized as follows:

- [Protecting a Website](#). This section gives a conceptual description of queuing via the Queue-it solution and presents the different ways internet visitors can enter a Queue-it waiting room.
- [Enter Waiting Room](#) Concept. Queue-it offers client-side and server-side integration options. These integration methods are described in this section.
- [Integration Configuration Details](#). The configuration of how, when, and where the waiting room shall protect the website.
- [Known User Integration Method Details](#). A server-side integration method.
- [JavaScript Integration Method Details](#). A client-side integration method.



- [Link To Waiting Room Integration Method](#) Details. A simple URL integration method.
- [Queue Skipping](#). Covers two ways to skip a queue.
- [Bot Management](#). Covers mechanism to prevent bots from interfering with the waiting room and ensure queuing fairness.
- [Visibility Modes](#). Covers the circumstances under which the waiting room will be visible to visitors.
- [Custom Theme](#). The visual representation of the visitor facing pages of the waiting room system.
- [Widgets](#). Add inline waiting room functionality to a website using HTML and JavaScript.
- [Queue Outflow](#). A flow-based approach and is designed so outflow follows redirect speed in any 60 second interval.
- [Queue-it Availability](#). Uptime and availability for service.

#### 1.4.1 Note on Terminology (as of March 2021)

As of October 2020 we have begun a terminology refresh. Several concepts have been given new names or redefined. Most notably, the term “Event” has been replaced with “Waiting room”. The term “Event” is visible many places, such as in code snippets where the term eventid is widely used.

We will continue to document terminology updates here on an ongoing basis.

Old Term	New Term	Comments
queue	waiting room	“waiting room” has replaced “queue” unless referring specifically to the line of users. The entity within the virtual waiting room is the waiting room.
event	waiting room	“waiting room” has replaced “event”. The entity within the virtual waiting room is the waiting room.
queue page	waiting room (page)	“waiting room” or “waiting room page” has replaced “queue page” to get away from “queue” terminology.
user	visitor	“visitor” has replaced “user” to sound more natural.
mixed mode	visible at peak	“visible at peak” has replaced “mixed mode” as part of an overhaul of visibility modes.

layout	theme	The look and feel of the waiting room.
--------	-------	--

## 1.5 Related Information

Additional documents include:

- Queue-it Security Considerations White Paper—This document covers customization and configuration of other aspects of the waiting room.
- Queue-it Load Test White Paper—This document summarizes the purpose and provides an overview of Load Testing.
- Queue-it Notifications and Logs White Paper—This document summarizes features and functionality of notifications.
- Queue-it Bots and Abuse Management White Paper —This document discusses how to secure the user flow, as well as to allowing only legitimate users to access the protected website.

These White Papers are available in the GO Queue-it Platform (**Help | White Papers and Guides**)

If you have feedback regarding the contents of this White Paper, please reach out to [Queue-it support](#).

## 2 Protecting a Website

This section gives a conceptual description of queuing via the Queue-it solution and presents the different ways visitors can enter a Queue-it waiting room.

### 2.1 The User Journey

By adding Queue-it to a website, visitors exceeding the website capacity limits are offloaded to the waiting room. Queue-it redirects the visitors who waited in line back to the website in the correct, sequential order.

Queue-it is not a proxy, and no data between the visitor and the target web site goes through the Queue-it servers.

### 2.2 Controlling the Waiting Room

The virtual waiting room solution provides a means to control and monitor traffic with real-time analytics by using the GO Queue-it Platform. It is used to manage queue outflow (how many visitors are allowed back into the site per minute), update waiting room state (running or paused) and communicate real-time updates to the visitors while they are in the waiting room.

### 2.3 The Waiting room Life-Cycle

A waiting room goes through different phases throughout its lifetime. Depending on which phase the waiting room is in the visitor experiences will vary. The phases are shown below.

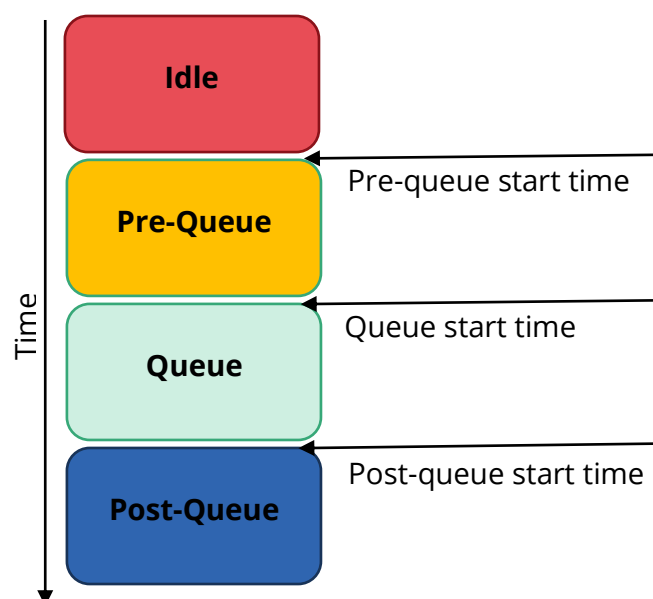


Figure 1: Waiting room Life Cycle

• Phase	• Explanation
Idle*	<p>This is the initial phase when the waiting room is not yet active. This means visitors visiting the site protected by a waiting room will not be enqueued (no queueid or queue number). Different options are configurable at this point. These include showing a special-, target- or before page to the visitors.</p> <p>The phase starts as soon as the waiting room is created and ends when the pre-queue phase starts.</p>
Pre-Queue*	<p>In the second phase visitors visiting the website protected by the waiting room will get enqueued. Visitors received a queueid, but not a queue number until just before waiting room start. The visitors will be presented with the pre-queue view.</p> <p>The length of this phase is controlled by the "Lifecycle Start Method" options on waiting room settings. Example: If the waiting room start time is 10:00, and the "Minutes before waiting room start" is set to 60, then the pre-queue phase will start at 9:00.</p>
Queue	<p>This is the primary phase where both queueids and queue numbers are issued to new and existing visitors. If the Visibility Mode is Always Visible or if visitors are waiting in the queue the visitors are shown the waiting room page. If Visibility Mode is At Peaks and the actual amount of visitors accessing the website can be accommodated within the max outflow which is allowed per minute, visitors will be redirected to the target site without seeing the waiting room.</p> <p>This phase ends when the post-queue start time is reached.</p>
Post-Queue	<p>This final phase is for latecomers to the waiting room or those visitors who are still in the queue when the post queue phase starts. They are presented with either the after-, special-, or target page.</p> <p>This phase starts at the end of the time of queue phase and this phase has no end time.</p>

*\*Idle and Pre-queue phases can be skipped if the waiting room is set to start immediately.*

## 3 Enter Waiting Room Concept

Queue-it offers client-side and server-side integration options. These Integration methods are described in the below sections and are further detailed in sections [4 \(Integration Configuration Details\)](#), [5 \(Known User Integration Method Details\)](#), and [6 \(JavaScript Integration Method Details\)](#).

The actual method for entering queues is controlled by the Integration Configuration described below. This uses the different Integration Methods described in [section 3.2 \(Different Integration Methods\)](#).

### 3.1 Integration Configuration

The Integration Configuration Section is inspired by Google's Tag Manager. Using the GO Queue-it Platform, one simple tool maintains all waiting rooms and configurations. After an initial Connector implementation, the GO Queue-it Platform is used to create, configure and modify waiting room behaviors. By using Actions, Triggers and Configuration, waiting room behaviors are configured to satisfy the waiting room needs within visitor journeys.

With this dynamic Integration Configuration, it is possible to update the behavior of the waiting room without the need to update the code on the website.

For more detail, see [section 4 \(Integration Configuration Details\)](#).

#### 3.1.1 Actions, Triggers and Configuration

- Actions are used to bind Triggers and Configuration settings to a running waiting room
- Triggers are used to initiate actions in accordance with the queuing logic of a site
- Configuration are used to set the cookie information

#### 3.1.2 Publish

When Actions, Triggers and Configuration are created and modified, these changes need to be Published through the GO Queue-it Platform. This allows the target site to pick up the latest configurations when it contacts the Queue-it servers.

### 3.2 Different Integration Methods

- **Known User:** Known User integration is the most solid and secure, as it is executed on the server side. The Known User integration can normally be done in one day by a skilled developer.
- **Queue-it JavaScript:** JavaScript integration is easier and faster to implement but not as secure, as it is executed by the client browser on relevant pages that are to be protected by the queue.

- **Link to Queue:** Visitors are directed to the waiting room.
- **Combined queue concepts:** It is also possible to combine Integration methods, such as using link to waiting room with JavaScript or using link to waiting room with Known User.

The Known User and the JavaScript integrations methods can be controlled by the setup of Integration Configuration described below.

### 3.2.1 Known User

Known User integration is the most solid and secure method, as it is handled on the server side. In the code executed by the server, an internal call to Queue-it verifies the visitors waiting room status. When the visitor is redirected back to the original website from the waiting room, a query string parameter is appended to the URL. This is used by the server-side code to verify that the visitor has been redirected from the waiting room. A shared secret token is used to verify the visitor.

It also directs the visitor before the execution of code in the browser, avoiding any previews of the target page on slow connections.

For more detail, see [section 5 \(Known User Integration Method Details\)](#).

### 3.2.2 Queue-it JavaScript

JavaScript is a client-side integration where the JavaScript code is inserted in the html <head>-section of all pages that are to be protected by the waiting room.

JavaScript integration is fast and easy to implement as it is handled on the client-side. As JavaScript is executed client-side, it can always be manipulated by a skilled person, making it possible to skip a waiting room. If there is a need to be sure that visitors cannot skip waiting room, we recommend implementation of the “Known User” back-end code.

For more detail, see [section 6 \(JavaScript Integration Method Details\)](#).

### 3.2.3 Link To Queue

Visitors enter the waiting room by being redirected to Queue-it URL. It can be done with a simple HTML <a>-tag and/or using code.

For more details, see [section 7 \(Link To Waiting Room Integration Method Details\)](#).

## 3.3 Combining Two Integration Methods

It is possible to combine a general JavaScript protection or Known User secured protection of a website with one or more linked waiting rooms. This could be relevant, for example, in a campaign situation where the JavaScript or Known User

is used to protect the entire site and at the same time one or more campaign waiting rooms are holding visitors waiting for a special promotion.

### 3.3.1 JavaScript and Link to Waiting Rooms

If visitors from the linked waiting room are sent to a normal webpage it is important that this page also includes the waiting room JavaScript. If not, visitors will be rejected once they hit the first page with the waiting room JavaScript.



Figure 2: Combining a JavaScript and Link Queues

### 3.3.2 Known User and Link to Queue

If visitors from the link to waiting room are sent to a normal webpage then it is important that Known User SDK protects the page.

When combining the two integration methods it is important to configure the link to waiting rooms with Known User Security and select "V3" and use the default secret key from the Security setup page.



Figure 3: Combining a Known User and Link to Waiting Rooms

## 4 Integration Configuration Details

The integration with Queue-it consists of two parts:

1. The **technical integration** which is done via client-side JavaScript or server-side code.
2. The **configuration** of how, when, and where the waiting room must protect the website.

The practical configuration is done via Triggers, Actions and Configurations:

1. Triggers are used to initiate Actions in accordance with the queuing logic of a site. They can pinpoint a specific item or cover a general area on a website.
2. Actions are used to bind Triggers and Configuration settings to a running waiting room.
3. Configuration settings are used to control the visitor session

Before diving into the configuration, the first step is to identify how many waiting rooms to use, where to place them, and when to open them. To do this first look at the infrastructure behind the website(s) to be protected.

### 4.1 Defining the right number of waiting rooms

When identifying the number of waiting rooms needed to protect a given system three elements need to be considered:

1. **Infrastructure:** In general, there must be one waiting room per system. For example, if there is an English website and a German website which both run on separate infrastructures then two waiting rooms running in parallel should be used. If, on the other hand, the two websites share the same infrastructure component, a database or web server for example, then normally only one waiting room should be used.
2. **Capacity of the various parts of the user journey:** Normally the static pages of a website will be able to handle much more traffic than the dynamic / personalized pages. It can therefore make sense to run two waiting rooms in sequence – one high speed waiting room in front of the static pages and a lower speed waiting room in front of the dynamic parts of the system, such as check-out and payment.
3. **Business requirements:** The business requirements can be more varied; below are some uses cases to illustrate how multiple waiting rooms can be used to support the business.

#### 4.1.1 Use case 1 – Localized Campaigns

An online retailer has 5 localized websites running on the same back-end. The business wants to run a campaign in the UK only. To avoid visitors from the 4 other



localities being queued for a local UK event, one waiting room is placed in front of the UK site and another in front of the 4 other sites. In this way it is possible to prioritize the system capacity between the UK campaign and the 4 other sites.

To configure the queues for this scenario:

1. On **Manage | Waiting Room** create two new waiting rooms (e.g. *ukshopwaitingroom* and *othershopwaitingroom*)
2. On **Integration | Triggers** create two new Triggers:
  - a. *UK trigger* with hostname Equals (ignore case) myshop.co.uk
  - b. *Other websites trigger* with hostname NotEquals (ignore case) myshop.co.uk
3. On **Integration | Actions** create two new Actions:
  - a. UK action combining the ukshopwaitingroom with the UK trigger
  - b. Other website action combining the otheshopwaitingroom with the Other website trigger
4. On **Integration | Overview** publish the configuration

#### 4.1.2 Use case 2 – One waiting room with multiple brands / languages

In the case where one system (same infrastructure) is running different brands (e.g. super-shoes.com and nice-shoes.com) or different languages (e.g. myshop.de and myshop.co.uk) the waiting room integration can support both a branding and a geographical dimension.

For each dimension make a Trigger (e.g. hostname Equals (ignore case) super-shoes.com) and use it on an Action where the corresponding Custom theme is specified. Use the same waiting room on all Trigger/Action pairs.

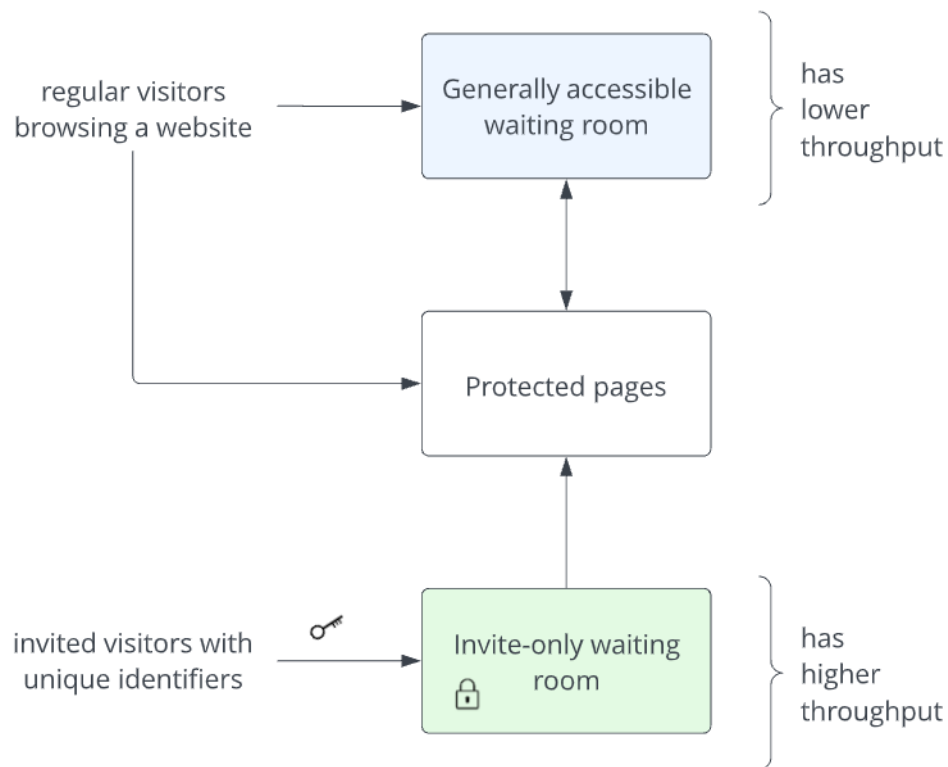
#### 4.1.3 Use case 3 – Differentiate between regular and invited visitors

Queue-it's invite-only waiting room makes it easy for customers to strictly regulate who can enter a waiting room.

You can use a list of unique visitor identifiers to specify who should have access to the waiting room. Visitors access the waiting room by using a link that contains the unique identifier. You can distribute these links via email or make the links available only once visitors have logged into their profiles.

You can run such a waiting room simultaneously with a generally accessible waiting room. All the visitors that do not have access links will be directed to the waiting room for regular visitors when they try to access a protected page.

The invite-only waiting room can be throttles independently of the waiting room for regular visitors.



**Figure 4: The flow for regular and invited visitors**

If you want to learn more about the “Invite-only” feature or if you need help setting it up, please contact Queue-it support at [support@queue-it.net](mailto:support@queue-it.net).

#### 4.1.4 Use case 4 – Protecting Specific Waiting Room Pages

In the case of a product launch or selling tickets for a special event, it is possible to set up a special waiting room protecting just those pages where this special product / event is sold. In this case visitors shopping for something else can either be sent to the same waiting room, but with another theme of the waiting room page -or- they can be sent to another waiting room.

To use one waiting room, but show different themes:

1. Select **Manage | Waiting Rooms** and create a new waiting room (e.g. *allvisitorswaitingroom*)
2. Select **Manage | Custom themes** and create two themes
  - a. A generic theme (e.g. *generictheme*)
  - b. A special xyz waiting room theme (e.g. *xyztheme*)
3. Select **Integration | Triggers** and create two Triggers
  - a. *Xyz waitingroom trigger* with PageUrl contains (ignore case) / waitingroom/xyz
  - b. Other waitingrooms trigger with

- i. PageUrl Contains (ignore case) /waitingroom
  - ii. AND PageUrl NotContains (ignore case) /xyz
- 4. Select **Integration | Action** and create two Actions
  - a. Xyz waitingroom action combining the alluserswaitingroom with the Xyz waitingroom trigger and using xyztheme
  - b. Other waitingroom action combining the allvisitorswaitingroom with the Other waitingroom trigger and using generictheme
- 5. Select **Integration | Overview** and publish the configuration

To separate visitors into two waiting rooms with different themes do as above but create two waiting rooms in step 1 and specify different waiting rooms in step 4.

Normally, 1 to 3 concurrent waiting rooms are enough, but there can be use cases where more are needed. A rule of thumb is that 5-8 waiting rooms are manageable, but beyond that the complexity of controlling them outweigh the benefits. Also, it is important to consider where all the traffic is hosted. If all waiting rooms are protecting pages that are all served on the same server, you need to split the volume between the multiple waiting rooms. If you can run a waiting room with a maximum outflow of 100 visitors per minute but you need to use two waiting rooms, you need to divide that 100 by two. In that case, the primary event could have a maximum outflow of 70 visitors per minute while the second, lesser prioritized event would have 30 visitors per minute. The key factor is to remember that your server capacity must be split between all events running on that server. If you do split these resources, it is possible to underutilize your server. If one waiting room is at capacity but the other is not, then the underutilized waiting room will be “wasting” waiting room spaces that the at capacity waiting room could be using.

#### 4.1.5 Planning for Events / Campaigns / Flash Sales

There are basically two ways to open a sale – and it’s all down to communication.

- 1. An **opening scenario**: Visitors have been told up front that the sale / event will start at a specific time in the future and if the visitors enter the website prior to that time, it is not possible to purchase / register.
- 2. A **soft-launch scenario**: The products are made available on the website first and the visitors are told about it through e-mail, social media or TV ads.

The difference here is that in the first scenario visitors will accumulate on the website before the sale / event starts and they will stay there (refreshing the page again and again) until the product they came for is available.

To handle the opening scenario, go to **Manage | Waiting Rooms** and create a waiting room with the start time matching the time where the sale starts and with a pre-queue time of e.g. 1 hour.

In general, the recommended length of the pre-queue is between ½ and 2 hours. If it is very short, there is a risk of the website becoming overloaded before the pre-

queue opens and visitors are off-loaded to the waiting room. If it is very long, there is a risk that a high number of visitors will take a place in the waiting room but will not show up when it's their turn many hours later.

In the **opening scenario** it is possible to set up the waiting room well before the event starts to avoid the stress of last-minute changes. This is controlled by the Idle and Pre-Queue phases.

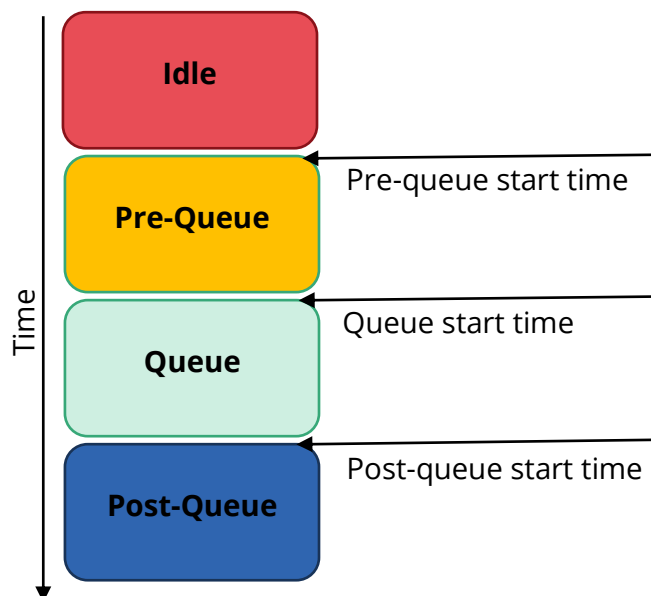


Figure 5: Waiting room Life Cycle

In the Idle phase the visitor can either be held on the pre-queue page, sent to a special page or (the most common option) allowed into the website until the Pre-Queue phase kicks in.

## 4.2 Controlling the Visitor Session

Once the visitor is through the waiting room and is redirected back to the website a session cookie is created. The cookie contains information about which waiting room the visitor has been through and the length of time the visitor can stay on the website before the session ends. A detailed description of the format and content can be found in next section.

The visitor session is configured under **Integration | Configuration** and specifies how long the session is valid in minutes and if it should be extended on each page request or not. On the individual Action's "Advanced" tab it is possible to overwrite the session cookie validity and extension policy specified in the configuration.

If the waiting room is disabled (controlled on the **Monitoring | Active Waiting Rooms**) the cookie validity is the same as when running (normal waiting room execution). When a waiting room is disabled visitors are not validated against the

Max Outflow and no metrics are recorded – It is like the waiting room was not there.

When the waiting room is in the Idle Phase and the Idle Logic is set to “Target URL” the cookie TTL is 3 minutes non-extendable. This is to ensure that all visitors that arrived at the website during the Idle phase are all sent to the pre-queue page.

#### 4.2.1 Session Cookie Format and Content

The visitor session cookie saved by either the JavaScript or the Known User SDK is named as `QueueITAccepted-SDFrts345E-V3_[EventId]`, where the [EventId] is the ID of the waiting room where the session is valid.

The cookie contains the following information:

QueueId	Visitors Unique ID from Queue
IssueTime	A UNIX timestamp of when the visitor's sessions was created / last updated (updated for KnownUser only, and used to determine the visitor session expiry on the server side)
Hash	A hash of the cookie content to ensure that visitors cannot tamper with the content of the cookie.
EventId	The ID of the waiting room where the session is valid.
RedirectType	Indicates the way a visitor was redirected to the Queue. See: <a href="#">RedirectType enum</a>

#### 4.2.2 User Journey Crosses Multiple Sub Domains

By default, the session cookie is saved on the domain where it is issued, so if the visitor's session starts on [www.myshop.com](http://www.myshop.com) that is where the cookie will be saved.

If the visitor journey continues to a different subdomain (for example [secure.myshop.com](http://secure.myshop.com)) the session cookie must be stored on the root domain ([myshop.com](http://myshop.com)). To achieve this, specify the root domain *myshop.com* in the **Integration | Configuration**.

#### 4.2.3 Canceling the Visitor Session

If there is a business need to cause the visitor to go back into the waiting room after a transaction or specific task, the cancel action will cancel a visitor's queueId. For example, visitors may make a single purchase, but upon purchase completion they should be forced back into the waiting room, at the end of the line.

To do this:

1. On the **Integration | Trigger** page add a new trigger matching the page shown after the visitor's purchase is 100% completed – usually on a confirmation or “thank you” page.
2. On the **Integration | Actions** page add a new Cancel Action with the Trigger specified above.
3. On the **Integration | Overview** Publish the changes.

This will clear the visitor's session both on the domain of the website and on the Queue-it domain. Be sure the order of the Actions logically follows, as they are followed in order, and the first Action found relevant will be the only Action run, see next section for more info.

## 4.3 Setting Up Multiple Actions

These are a few things to be aware of when setting up multiple Actions pointing to different waiting rooms.

### 4.3.1 The Order of Actions Matters

When configuring more than one Action the order matters. Actions are evaluated in the order specified on **Integration | Actions** from the top down.

In general, the more specific Actions should be at the top and the more generic catch-all type lower in the list.

### 4.3.2 The “All Pages” Trigger

There is a built-in Trigger called “All Pages (Queue-it)”. This Trigger is handy if the goal is a simple catch-all waiting room across all pages.

Sometimes a combination of a low-speed waiting room protecting the check-out flow with a high-speed At Peaks waiting room can be useful. If the “All Pages (Queue-it)” Trigger is used for the At Peaks waiting room, visitors can end up in a situation where they first are queued in the catch-all high-speed At Peaks waiting room (because inflow is above max) and then when they reach the check-out flow, they queue again on the second waiting room.

This is not bad, as the two waiting rooms are protecting different parts of the website, they have different capacity.

## 4.4 Handling e-mail / Facebook / Twitter Campaigns (Link to Waiting Room)

Sending campaign e-mails to many people or posting a link on social media can generate a large spike in traffic (inflow) to a website. To avoid the initial hit of visitors requesting the initial page before the Queue-it integration sends the visitors to the waiting room, a link directly to the waiting room can be used instead.

Please refer to [section 7 \(Link To Waiting Room Integration Method Details\)](#) for details on how to use the Link to Queue.

## 4.5 Handling Call-backs from Payment Provider

Often the payment provider uses a call back to the website to complete the payment transaction. To avoid these payment call-backs getting queued up add the call-back URL to the Trigger with an AND PageUrl NotContains (ignore case) /PART OF CALL-BACK URL/

For PayPal this could e.g. be: /payment\_paypal/

## 4.6 Trigger Types

These Trigger types are currently supported:

Name	Supported Integration Methods	Comments
URL	JavaScript + Known User	Supports PageUrl, HostName and PagePath. Take <a href="https://mydomain.com/shop/cars.aspx">https://mydomain.com/shop/cars.aspx</a> as an example: - <a href="https://mydomain.com/shop/cars.aspx">https://mydomain.com/shop/cars.aspx</a> is the PageUrl - mydomain.com is the HostName - /shop/cars.aspx is the PagePath
JavaScript	JavaScript	Can be used to trigger on the value of a named JavaScript variable *) on the page.
Cookie	JavaScript + Known User	Can be used to trigger on the value of a named cookie on the page.
UserAgent	JavaScript + Known User	Can be used to trigger on the value of the requests User Agent string.
HTTPHeader	Known User	Can be used to trigger on the value of the requests HTTP Header string.

Observe that the name of the JavaScript variable and cookie are **case-sensitive**.

If the name of the JavaScript variable or cookie is not found (either because it is not defined, or it is misspelled) then the Equals and Contains operator will always return false.

## 4.7 Action Types

Actions play an important role in combining the waiting room with the Triggers and the Configuration. If just one of the selected Triggers are matched for the referenced waiting room then the waiting room will fire.

It is very important to note that if you have multiple Actions then order of the Actions matters when the triggers are not mutually exclusive or if they have the same triggers e.g. All Pages (Queue-it).

Example : If your first action is 'ABCTest01' and your second action is 'XYZtest01' but they have the same trigger like All Pages (Queue-it), then the first action will always be executed which is 'ABCTest01' and the second Action 'XYZtest01' will never be triggered as the Triggers are not mutually exclusive.

The following Action types are supported in both a JavaScript and a Known User version:

Name	Function
Queue	Sends the visitors to the specified waiting room with the specified Custom theme, Language and Configuration.
Cancel	Deletes the visitors Queue-it session cookie on both the customers domain and on the Queue-it domain.
Ignore	Take no action. This can be useful if all except a few pages should be protected. Starting with an Ignore Action on the few pages followed by a Queue Action on all pages will achieve this.

### 4.7.1 Queue Action

On the Basic Tab it is possible to overwrite the settings on the waiting room. This is used if the same waiting room is used to protect two differently branded websites and the waiting room should have the same skin / brand as the website sending the visitor to the waiting room – see the different use cases in [section 4.1 \(Defining the right number of \)](#) for more info.

The Basic tab is also the place to specify which Triggers should be used for the referenced waiting room. The Triggers are OR'ed, so whenever one of these triggers is matched for this action then the Queue Action is used.

On the Advanced tab it is possible to control:

- **Cookie Validity Time:** This field is optional. It is the validity of the session cookie in minutes. If it is not specified, then the value selected under configuration will be used
- **Extend Cookie Validity:** By default, the behavior specified on the configuration is used, but it can be overwritten here



- Redirect Logic: Customer can change the default behavior where visitors are redirected back to the URL where they entered the waiting room (Origin). Alternatives are:

Waiting room target URL: force all visitors to the URL specified as the target URL on the waiting room

Alternate page: forces all visitors to an alternate page

#### 4.7.2 Cancel Action

Some customers would like to Cancel the queue of a visitor after he or she has made the purchase. This functionality will keep the visitor from buying more than one time. If the visitor would like to buy more then he or she will have to wait again in the waiting room.

The Cancel functionality basically cancels the queue and flushes cookies and related information of a visitor for the defined trigger for the Cancel Action. Once cancelled, the system will treat the same visitor as a new visitor as if he or she started browsing the website for the first time.

However, it is very important to make sure that the visitor journey is completed before invoking the cancel feature (e.g. Triggering the Cancel Action on the final Page like the Thank you page), as it will place the visitor at the end of queue.

#### 4.7.3 Ignore Action

The Ignore action simply ignores queuing if the trigger has been met. This way bypasses can be set up with a single Action. This can for example be used for the following use cases:

- Bypass all good bot traffic (see details on bot management in [section 9](#))
- Allow single servers to skip the waiting room, e.g. with a specific IP-address in a header trigger
- Allow a reverse proxy to skip the waiting room, e.g. with a special header trigger

### 4.8 Configurations

The more technical parts of the integration are specified in **Integration | Configurations**. The "Default (Queue-it)" configuration will save the session cookie on the domain of the current page with a validity time of 20 minutes that will get extended on each new page request.

If needed, custom configurations can be created and used on one or more Actions.

For more info on how to control the visitor's session please refer to [section 4.2 \(Controlling the Visitor Session\)](#).

#### 4.8.1 Special for the JavaScript Integration

When the visitor is through the waiting room, the URL contains query string parameters that allows Queue-it to verify that the visitor has been through the waiting room. For example:

<http://secure.cuppercakes.com:4433/BuyCakes.aspx?q=cbaecaa9-b497-4491-9812-727f9c8960cc&p=8d143487-8418-45d3-b225-215d9108454b&ts=1477637528&c=ticketania&e=cuppercakes05&rt=Queue&h=e5849b4d77b6a894381f088c51f75667>

This token is personal and will expire so it should not be bookmarked or shared. The token can be removed from the URL by checking the “Remove queue-it params” checkbox.

For Known User integrations this is handled in code on the server side.

## 5 Known User Integration Method Details

The Known User implementation method tests if a specific visitor has arrived at the protected site via the waiting room, or if the visitor has attempted to skip the waiting room. By adding a server-side integration to the protected server, the Queue-it security framework ensures that visitors cannot bypass the waiting room. When a visitor is redirected back from the waiting room to the protected website, the waiting room engine will attach a query string parameter (*queueittoken*) containing information about the visitor which is hashed to prevent tampering.

### 5.1.1 Known User Flow

The Known User logic is described in the following steps and diagram.

1. A visitor requests a page on the protected server
2. The server-side validation method determines that the visitor has no Queue-it session cookie and no *queueittoken*. It then sends the visitor to the correct waiting room based on the integration configuration (see section 3.1 )
3. Visitor waits in the waiting room until his or her turn comes up
4. Visitor is redirected back to the protected website, now with a *queueittoken*
5. The server-side validation method validates the *queueittoken* and creates a Queue-it session cookie
6. The visitor browses to a new page and the existence of the Queue-it session cookie prevents him or her from queuing again

## KnownUser validation

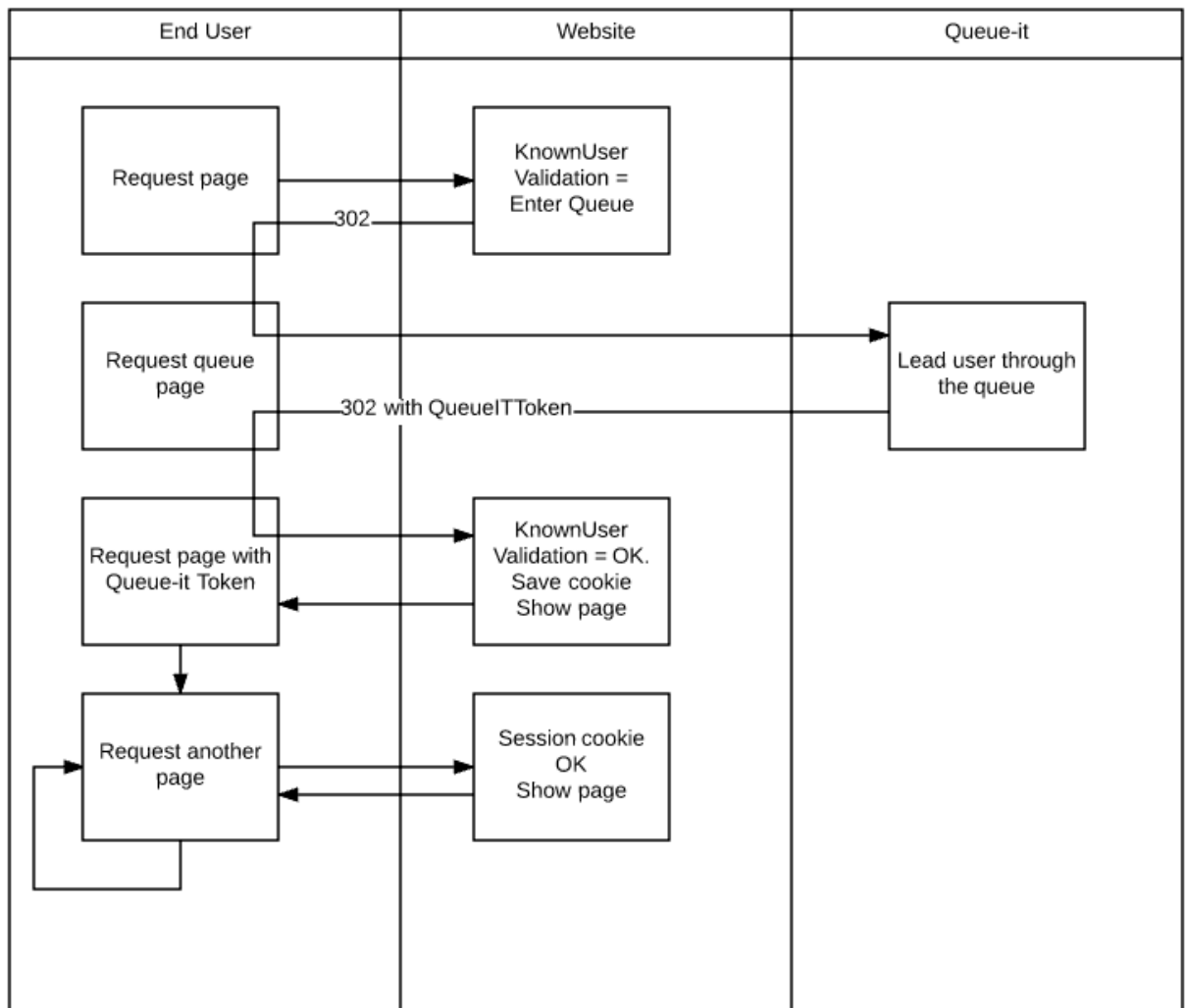


Figure 6: Known User Validation

### 5.1.2 Supported Languages

- ASP.NET
- PHP
- Java
- RubyOnRails
- Node.js
- Android
- iOS

The SDKs for these languages are open sourced and can be found here:

<https://github.com/queueit>

For other programming languages like Classic ASP and Cold Fusion, please contact Queue-it support at [support@queue-it.net](mailto:support@queue-it.net).

### 5.1.3 Configuring the Waiting Room(s)

The configuration for protecting the website is separated from the validation logic. This ensures that changes to the behavior of the waiting room can be implemented without the need to redeploy code to the website.

The configuration for behaviors can be consumed by the Known User SDK in two ways:

1. By using the GO Queue-it Platform to load the configuration
2. By providing the configuration as an object in code

Option one is recommended in most cases, as it is usually the simplest solution. It also gives Queue-it support a better way to understand and support each specific use case. If there is a need to deploy new waiting room daily or weekly with a tight integration into the own back-end systems, option two might be the right method together with Queue-it API.

The configuration in option one specifies a set of Triggers and Actions. A Trigger is an expression matching a single, multiple, or all the URLs on a website. When a visitor enters the protected website and the URL matches a Trigger-expression the corresponding Action will be invoked. The Action specifies which waiting room the visitors should be sent to. In this way, changes to the behavior of the waiting room can be modified without the need to redeploy code to the website.

For more information about how to setup Triggers and Actions, please look at [section 3.1.1 \(Actions, Triggers and Configuration\)](#).

## 5.2 Know User Code flow

Known User code flow follows the following steps to validate that the current visitor may enter the protected website (has been through the waiting room).

1. Read the waiting room configuration (stored locally or on Queue-it servers)
2. Validate the *queueittoken* and store a session cookie
3. Redirect the visitor

### 5.2.1 Read the waiting room configuration

If the GO Queue-it Platform is used to configure the behavior of the waiting room (option one in [section 5.1.3 \(Configuring the Waiting Room\(s\)\)](#)) then please refer to [section 5.3.1 \(Accessing Configuration Via the GO Queue-it Platform\)](#) to configure section of this document on how to download the configuration to the protected website.

If the configuration is done in code (option two in [section 5.1.3 \(Configuring the Waiting Room\(s\)\)](#)) please refer to [section 5.3.2 \(Accessing the Configuration as an Object in Code\)](#) for details.

### 5.2.2 Validate the *queueit* token and Store a Session Cookie

To validate that the visitor has been through the waiting room, use the `KnownUser.ValidateRequestByIntegrationConfig()` method. This call validates the timestamp and hash and if they are valid creates a "QueueITAccepted-SDFrts345E-V3\_[EventId]" cookie with a TTL as specified in the configuration.

In some programming languages such as .NET, Java and PHP cookies are handled in a generic way and the Known User SDK handles it automatically. In other languages like node.js, different cookie middleware exists, and adjustments might need to be made to accommodate for that.

More information on configuring the session cookie can be found in [section 6.6 \(Queue-it Session Cookie\)](#).

### 5.2.3 Redirect the Visitor

The decision to redirect the visitor is made at this point. The result of the `ValidateRequestByIntegrationConfig()` call has a property called `DoRedirect`. If the value for this property is true, the visitor is redirected to the waiting room. If it is false, the visitor may stay on the requested page and will not be redirected.

The visitor can be redirected to the waiting room because it is the first time he or she requests a page protected by the waiting room, because the hash is invalid, or because the session has expired.

For more information on different error scenarios please refer to [section 5.5 \(Troubleshooting\)](#).

### 5.2.4 Rejection flow

If a visitor is rejected by the above Known User code, don't redirect them directly back into Queue-it, as it could lead to circular situations. Create a flow as follows:

1. Redirect the visitor to an Error page, describing the error (not too many details to prevent brute forcing). The Error page can have a link back to the waiting room.
2. Going back to the waiting room must be done via the Queue-it cancel page. This page clears the cookie on the visitors' browser, canceling the queue and then redirecting the visitor back to the waiting room.

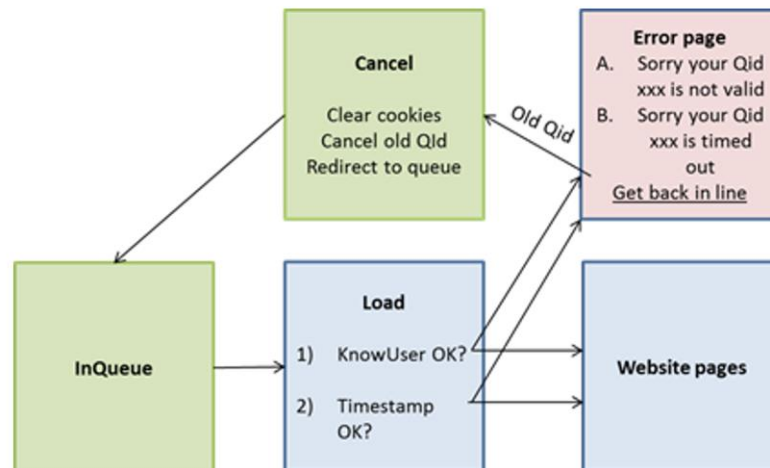


Figure 7: Rejection flow

Queue-it cancel page is called with c, e, q-parameters and an optional r-parameter. Using cancel page with c, e, and q-parameter will redirect the visitor back to Queue-it and request a new queue-number.

e.g.: <http://q.queue-it.net/cancel.aspx?c=ticketania&e=test12&q=e4131f6a-cb31-42ea-8bfe-30e5924e334b>

Using the &r=[redirectEncodedUrl] will redirect the visitor to the given encoded URL instead of directly to the queue, to follow the original flow once the queueID is cancelled. e.g.:

<http://q.queue-it.net/cancel.aspx?c=ticketania&e=test12&q=e4131f6a-cb31-42ea-8bfe-30e5924e334b&r=http%3A%2F%2Fwww.ticketania.com>

## 5.3 Known User Implementation

As described in [section 5.1.3](#) it is possible to implement the Known User validation in two ways:

1. Accessing configuration via the GO Queue-it Platform
2. Accessing the configuration as an object in code

The recommended method is to use the GO Queue-it Platform to set up the configuration. For detailed implementation in the preferred programming language please refer to GitHub: <https://github.com/queueit>.

### 5.3.1 Accessing Configuration Via the GO Queue-it Platform

The Known User validation must only be done on *page requests* which need to be queued. If the Known User validation logic is added to a central place (like global.asax in .NET), Triggers (waiting room request validation) must only fire on page requests and not on image or AJAX requests, for example.

### 5.3.1.1 Implementation Sample (ASP.NET C#)

This example is using the *IntegrationConfigProvider* (refer to the Queue configuration Downloader section in 5.3.4 for details) to download the waiting room configuration.

The following method is used to validate that a visitor has been through the waiting room:

```
private void DoValidation()
{
    try
    {
        var customerId = "Your Queue-it customer ID";
        var secretKey = "Your 72 char secrete key as specified in GO Queue-it self-service platform";
        var queueitToken = Request.QueryString[KnownUser.QueueITTokenKey];
        var pureURL = Regex.Replace(Request.Url.ToString(), @"([\?&]) (" + KnownUser.QueueITTokenKey
                                   + "=[^&]*)", string.Empty, RegexOptions.IgnoreCase);

        // The pureURL is used to match Triggers and as the Target URL
        // (where to return the users to)

        // It is therefore important that the pureURL is exactly the URL
        // of the users browsers.

        // So if the protected webserver is e.g. behind a load balancer
        // that modifies the host name or port,

        // reformat the pureURL before proceeding
        var integrationConfig = IntegrationConfigProvider
            .GetCachedIntegrationConfig(customerId);

        //Verify if the user has been through the queue
        var validationResult = KnownUser
            .ValidateRequestByIntegrationConfig(pureUrl, queueitToken,
                                                integrationConfig, customerId, secretKey);

        if (validationResult.DoRedirect)
        {
            //Adding no cache headers to prevent browsers to cache
            //requests
            Response.Cache.SetCacheability(HttpCacheability.NoCache);
            Response.Cache.SetExpires(DateTime.UtcNow.AddHours(-1));
            Response.Cache.SetNoStore();
            Response.Cache.SetMaxAge(new TimeSpan(0, 0, 30));

            //Send the user to the queue - either because hash was missing
            //or because is was invalid
            Response.Redirect(validationResult.RedirectUrl);
        }
    }
}
```



```

    }
    else
    {
        //Request can continue - we remove queueittoken from querystring
        parameter to avoid
        //sharing of user specific token

        if(HttpContext.Current.Request.Url.ToString().Contains(KnownUser.Queue
        ITTokenKey))
            Response.Redirect(pureUrl);
    }
}
catch (Exception ex)
{
    //There was an error validating the request
    //Use your own logging framework to log the Exception
    //This was a configuration exception, so we let the user continue
}
}

```

### 5.3.1.2 Alternative methods to access the configuration file

If the application server (maybe due to security reasons) is not allowed to do external GET requests, then there are two options:

1. Manually download the configuration file from the GO Queue-it Platform, save it on the application server and load it from local disk, by changing this line:  
var integrationConfig =  
IntegrationConfigProvider.GetCachedIntegrationConfig(customerId);
2. Use an internal gateway server to download the configuration file and save to application server

### 5.3.2 Accessing the Configuration as an Object in Code

To specify the configuration in code without using the Trigger/Action paradigm, a config object is parsed into the KnownUser.ValidateRequestByLocalEventConfig() method. Also, in this case it is important only to queue-up page requests and not requests for resources or AJAX calls.

#### 5.3.2.1 Implementation sample (ASP.NET C#)

The following is an example of how to specify the configuration in code:

```

private void DoValidationByLocalEventConfig ()
{
    try
    {

```

```

var customerId = "Your Queue-it customer ID";

var secretKey = "Your 72 char secret key as specified in the GO
Queue-it self-service platform";

var queueitToken = Request.QueryString[KnownUser.QueueITTokenKey];
var pureUrl = Regex.Replace(Request.Url.ToString(), @"([\?&])(" +
KnownUser.QueueITTokenKey
    + "=[^&]*)", string.Empty, RegexOptions.IgnoreCase);
var eventConfig = new EventConfig()
{
    EventId = "event1", //ID of the queue to use
    CookieDomain = "mydomain.com", //Optional - Domain name where
the Queue-it session cookie should be saved. Default is to save on the
domain of the request
    QueueDomain = "queue.mydomain.com", //Optional - Domain name of
the queue. Default is [CustomerId].queue-it.net
    CookieValidityMinute = 15, //Optional - Validity of the Queue-it
session cookie. Default is 10 minutes
    ExtendCookieValidity = false, //Optional - Should the Queue-it
session cookie validity time be extended each time the validation
runs? Default is true.
    Culture = "en-US", //Optional - Culture of the queue ticket
layout in the format specified here: https://msdn.microsoft.com/en-
us/library/ee825488\(v=cs.20\).aspx Default is to use what is specified
on Event
    LayoutName = "MyCustomLayoutName" //Optional - Name of the queue
ticket layout - e.g. "Default layout by Queue-it". Default is to use
what is specified on the Event
};

//Verify if the user has been through the queue
var validationResult =
KnownUser.ValidateRequestByLocalEventConfig(
    pureUrl, queueitToken, eventConfig,
customerId, secretKey);

if (validationResult.DoRedirect)
{
    //Adding no cache headers to prevent browsers to cache requests
    Response.Cache.SetCacheability(HttpCacheability.NoCache);
    Response.Cache.SetExpires(DateTime.UtcNow.AddHours(-1));
    Response.Cache.SetNoStore();
    Response.Cache.SetMaxAge(new TimeSpan(0, 0, 30));

    //Send the user to the queue - either because hash was missing
or because is was invalid

```

```

        Response.Redirect(validationResult.RedirectUrl);
    }
    else
    {
        //Request can continue - we remove queueittoken form querystring
        parameter to avoid sharing of user specific token

        if
        (HttpContext.Current.Request.Url.ToString().Contains(KnownUser.QueueIT
        TokenKey))
            Response.Redirect(pureUrl);
    }
}
catch (Exception ex)
{
    //There was an error validating the request
    //Please log the error and let user continue
}
}
}

```

### 5.3.3 pureUrl

The pureUrl is used to match Triggers and as the Target URL (where to return the visitors to). It is therefore important that the pureUrl is exactly the URL of the visitor's browsers. If the webserver is behind a load balancer that modifies the host name or port for example, reformat the pureUrl before proceeding.

If the waiting room is not triggered as expected verify that the pureUrl is exactly as seen in the browsers address field. Please refer [section 5.5 \(Troubleshooting\)](#) on how to troubleshoot the integration.

### 5.3.4 Queue Configuration Downloader

The Integration Configuration provider is responsible for downloading the latest configuration from the GO Queue-it Platform to the local infrastructure. The recommendation is to create a timer function that can download and cache the configuration periodically, for example every 5-10 minutes. The latest version of the configuration file can be found here: [https://\[your-customer-id\].queue-it.net/status/integrationconfig/secure/\[your-customer-id\]](https://[your-customer-id].queue-it.net/status/integrationconfig/secure/[your-customer-id]) after a successful publish in the integration section of the GO Queue-it Platform. **To prevent unauthorized access to your config file you must enable the "Secure integration config" setting in the GO Queue-it Platform** and then provide your API key in the request header to retrieve the integration config.

Please refer to GitHub repository for downloader example, for instance <https://github.com/queueit/KnownUser.V3.ASPNET/tree/master/Documentation>

The higher-level logic is as follows:

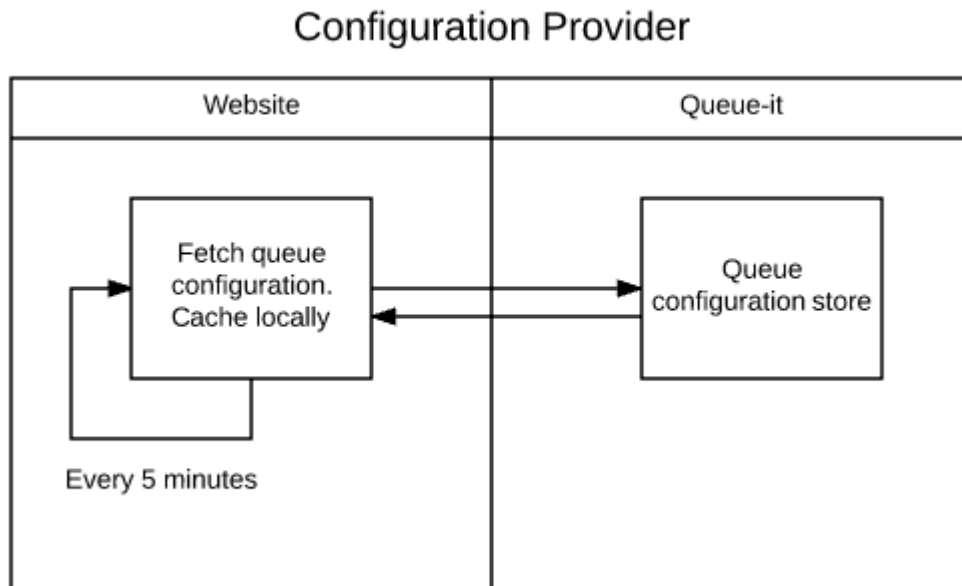


Figure 8: Configuration Provider

### 5.3.5 Query String Parameters (Queue-it Token)

When the Known User security is activated, all visitors redirected from Queue-it to the protected site get an extra query string parameter *queueittoken* with a set up values including a hash. This hash value is used in the verification and ensures that nobody has tampered with the query string parameter. A timestamp is also added.

The parameters contain the following:

- *q*: the visitors's unique ID, a GUID
- *e*: the waiting room ID
- *ts*: an integer timestamp counting number of seconds since 1970-01-01 00:00:00 UTC
- *h*: an integer calculated hash
- *rt*: string value that shows in what way the visitors were redirected
- *ec*: Cookie extendable

#### 5.3.5.1 *q* Parameter

The *q* parameter is a GUID with the visitors's unique ID, e.g. de03b295-7a47-4c7b-a222-b076c8a5f474. The ID is generated when the visitor access a queue-protected page.

The target system can store the ID together with a business transaction and hence has a reference to the ID.

### 5.3.5.2 *ts* Parameter

The *ts* parameter is generated by Queue-it when the visitor is redirected and is an integer timestamp counting the number of seconds since 1970-01-01 00:00:00 UTC (UNIX timestamp).

The Known User SDK will verify that this timestamp is no more than 3 minutes old; hence preventing the entire URL from being too old. This can for example prevent someone from sharing the target link on the internet.

It is very important that the Queue-it platform and the target platform are synchronized. The Queue-it platform is synchronized with the time.nist.gov Internet time on a regular interval.

### 5.3.5.3 *e* Parameter

The *e* parameter is the ID of the waiting room. The ID is created when a waiting room is created.

### 5.3.5.4 *rt* Parameter

The *rt* parameter shows where the uvisitor is coming from. These are the possible value:

- Queue: Visitor has been redirected to the target URL by the queue
- SafetyNet: Visitor has been redirected to the target URL by the SafetyNet (not queued in a peak protected waiting room)
- AfterEvent: Visitor has been redirected to the target URL after the waiting room has ended
- Disabled: Visitor has been redirected to the target URL while the waiting room was disabled
- DirectLink: Visitor has been redirected to the target URL using a direct link and has not been through the waiting room
- Idle: Visitor has been redirected to the target URL while the waiting room was idle

### 5.3.5.5 *ec* Parameter

The *ec* parameter indicates if the visitor's session cookie can be extended or not (taken from the **Integration | Configuration**)

### 5.3.5.6 *h* Parameter

The *h* parameter is an integer calculated hash value based on the queueittoken and secret key set on the waiting room (**Account | Security**).

As an example, the visitor will be redirected to:

[https://secure.ticketania.com/BuyTickets.aspx?queueittoken=e\\_tecketaniademo~q\\_9925866f-fa25-4f19-a4c6-22242b47cf5b~ts\\_1505204395~ce\\_True~rt\\_Queue~h\\_ada5083648c25b712584bc3bdf2972e5947fc2d2851b99c478c6b1b774b86a3f](https://secure.ticketania.com/BuyTickets.aspx?queueittoken=e_tecketaniademo~q_9925866f-fa25-4f19-a4c6-22242b47cf5b~ts_1505204395~ce_True~rt_Queue~h_ada5083648c25b712584bc3bdf2972e5947fc2d2851b99c478c6b1b774b86a3f)

<https://secure.ticketania.com/BuyTickets.aspx> is the URL to which Queue-it will redirect the visitor to when their turn is up. The queueittoken consists of the following parameters:

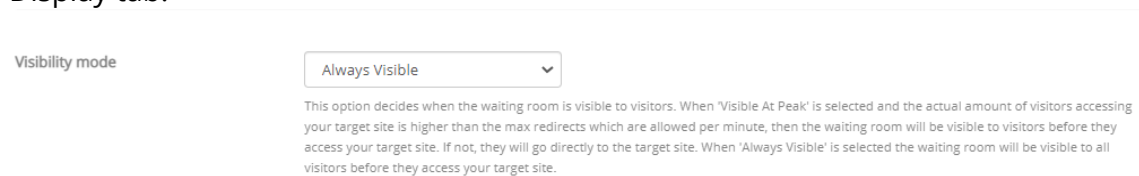
- q\_9925866f-fa25-4f19-a4c6-22242b47cf5b is the visitor's unique Queue ID (GUID)
- e\_tecketaniademo is the waiting room ID
- ts\_1505204395 is an integer timestamp counting number of seconds since 1970-01-01 00:00:00 UTC
- h\_ada5083648c25b712584bc3bdf2972e5947fc2d2851b99c478c6b1b774b86a3f is the calculated hash
- rt\_Queue represents the visitor is coming from the queue (Redirect type)

## 5.4 Known User Integration Test

To verify the Known User implementation, the following tests should be performed, at a minimum.

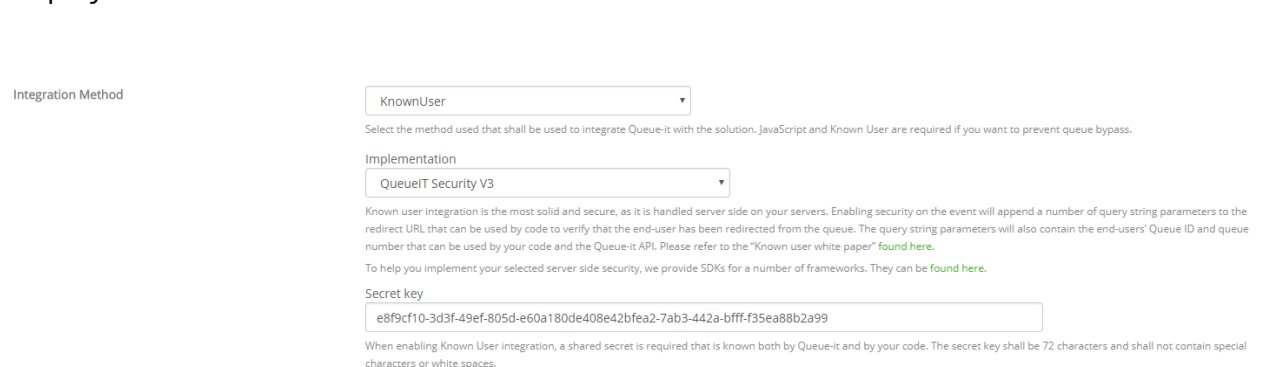
### 5.4.1 Prerequisites

To make the test easier, please ensure the test waiting room has been configured to always display queue numbers to visitors on the Waiting Room Settings page's Display tab.



**Figure 9: Display Waiting Room Page**

Ensure that Known User is set to "QueueIT Security V3" on the waiting room settings and that the "Known users' secret key" matches what has been configured in the Known User implementation on the Waiting Room Settings page's Deployment tab.



**Figure 10: Enable "Known User"**

This document uses our demo site for demonstration purposes. The Ticketania demo site consists of two parts:

1. A CMS site with all static pages (<http://www.ticketania.com>). This site is not protected by a waiting room.

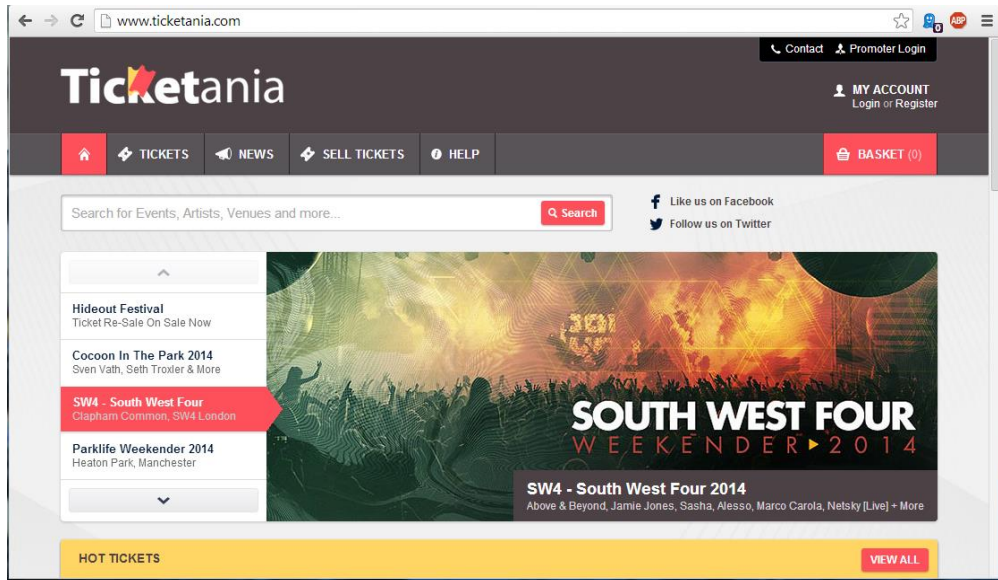


Figure 11: Ticketania Site

2. A transactional website (<https://secure.ticketania.com>) that is protected by Queue-it.

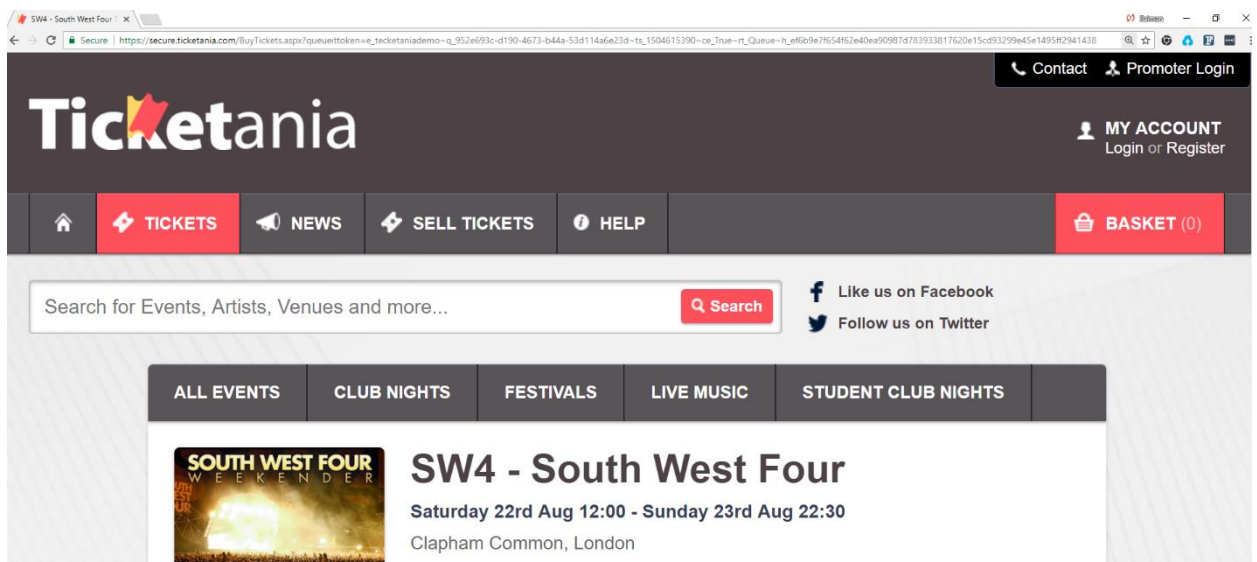


Figure 12: Ticketania Site

Both sites only contain a single web page.

## 5.4.2 Successful Flow

The first test is to verify that a successful flow is possible.

1. Close all Private browser windows.

2. Open a new Private browser window
3. Go to website e.g. <http://www.ticketania.com>
4. Open the Developer tool in the browser and enable the network trace
5. Click on the link to the protected website (banner with "SOUTH WEST FOUR" on demo site)
6. Verify the queue number is shown for approx. 20 seconds
7. Verify redirect to the protected website including the *queueittoken* query string parameter like this:  
[https://secure.ticketania.com/BuyTickets.aspx?queueittoken=e\\_tecketaniademo~q\\_c84f952d-fe5e-45ad-a114-624e45c9d0a3~ts\\_1504275185~ce\\_True~rt\\_Queue~h\\_a4a4b43a1558eb6a9ba709152079b1d7fbe14e63b92e21a2432147b46b2bca9b](https://secure.ticketania.com/BuyTickets.aspx?queueittoken=e_tecketaniademo~q_c84f952d-fe5e-45ad-a114-624e45c9d0a3~ts_1504275185~ce_True~rt_Queue~h_a4a4b43a1558eb6a9ba709152079b1d7fbe14e63b92e21a2432147b46b2bca9b)  
 This can be seen in the network trace.
8. Verify that the URL in the browser does not contain the *queueittoken* as it should be removed by the server-side implementation e.g.  
<https://secure.ticketania.com>
9. Verify that the entire purchase flow continues without being sent back to the waiting room.

### 5.4.3 No Looping Around

This test is to ensure that visitors are not looped around (sent back to the waiting room).

1. Close all Private browser windows.
2. Open a new Private browser window
3. Open a new Incognito (Chrome) / InPrivate (IE) browser
4. Open the browsers' development tool (press F12) and select "Network". Ensure to check the "Preserve log" check-box

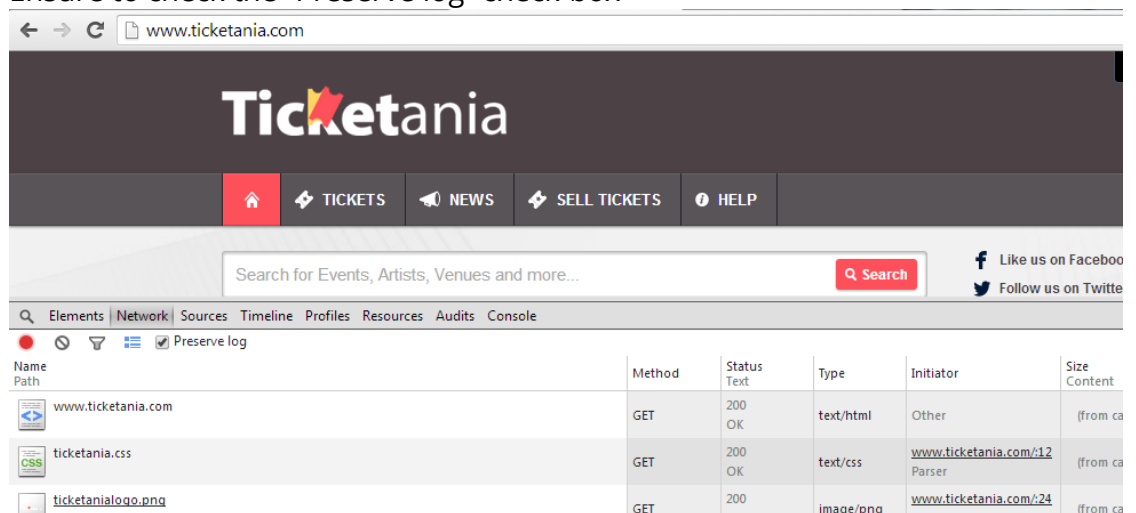


Figure 13: Browser Network Check



5. Go to the protected website e.g. <http://www.ticketania.com>
6. Click on the link to the protected website (banner with "SOUTH WEST FOUR" on demo site)
7. Verify the queue number for approx. 20 seconds
8. Verify the redirect to the protected website e.g. <https://secure.ticketania.com>
9. Inspect the network log and ensure that there is only one set of enqueue.aspx and ticket-shop-landing-page (BuyTickets.aspx in demo) in the log



	?c=ticketania&e=tecketaniademo&q=3caa6981-bac6-43b6-af30-5d0dfc095420&cid=en-US	GET	302
	BuyTickets.aspx?queueittoken=e_tecketaniademo~q_3c...7724644d9e3330be2658f0c2b5f878fe19745720ae9cd05ea	GET	200

Figure 14: Browser Network View

10. Verify that the ~q parameter in the two URLs match

#### 5.4.4 Tampered Link

This test is to ensure that visitors that try to tamper with the link from the waiting room and to your website are rejected.

1. Close all Private browser windows.
2. Open a new Private browser window
3. Open the developer tool in the browser and enable the network trace
4. Go to the protected website e.g. <http://www.ticketania.com>
5. Click on the link to the protected website (banner with "SOUTH WEST FOUR" on demo site)
6. Verify the queue number for approx. 20 seconds
7. Verify the redirect to the protected website e.g. <https://secure.ticketania.com>
8. Find the URL in the network trace that contains the *queueittoken* query string parameter
9. Copy the URL to e.g. Notepad
10. Close the Incognito browser again
11. Change one or more characters in the *queueittoken* in Notepad
12. Open a new Incognito browser and paste the URL from Notepad into the address field
13. Verify that you are shown an error page that explains that the queue number could not be validated and with a link back to re-enter the waiting room

### 5.4.5 Shared Link

This test is to ensure that links from the waiting room and to your website are only active for a specified time period to protect your website if a visitor shares such a link on e.g. Twitter or Facebook. The queue number expiration is 3 minutes.

1. Open a new Incognito (Chrome) / InPrivate (IE) browser
2. Open the developer tool in the browser and enable the network trace
3. Go to your website e.g. <http://www.ticketania.com>
4. Click on link to protected website (banner with "SOUTH WEST FOUR" on demo site)
5. Verify that you see the queue number for approx. 20 seconds
6. Verify that you get to the protected website e.g. <https://secure.ticketania.com>
7. Find the URL in the network trace that contains the *queueittoken* query string parameter
8. Copy the browsers' URL to e.g. Notepad
9. Close the Incognito browser again
10. Wait a bit longer than 3 minutes
11. Open a new Incognito browser and paste the URL from Notepad into the address field
12. Verify that you are shown an error page that explains that the queue number is too old and contains a link to re-enter the waiting room.

## 5.5 Troubleshooting

If the queue integration is working as expected in a development environment but is failing in a test / QA environment where live debugging is not possible, Queue-it can generate a debug link that will store some debug information in a cookie.

### 5.5.1 Debug link

The debug link can be used to verify that the waiting room configuration is properly configured and synchronized. If the Known User code is running on the page of the debug link, then it will save a cookie called 'queueitdebug' with the following information:

Field	Description
ConfigVersion	The version of the downloaded waiting room configuration used by the SDK

<b>PureUrl</b>	The purlUrl as seen by the SDK
<b>QueueitToken</b>	The QueueitToken passed to the SDK
<b>OriginalUrl</b>	The original URL of the debug page as seen by the SDK
<b>ServerUtcTime</b>	The server's UTC time
<b>RequestIP</b>	The clients IP as seen by the SDK
<b>RequestHttpHeader_Via</b>	The VIA general header if added by proxies and as seen by the SDK. <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Via">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Via</a>
<b>RequestHttpHeader_Forwarded</b>	The Forwarded HTTP header as seen by the SDK. <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/forwarded">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/forwarded</a>
<b>RequestHttpHeader_XForwardedFor</b>	The originating IP address of a client connecting to a web server through an HTTP proxy or a load balancer as seen by the SDK. <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-For</a>
<b>RequestHttpHeader_XForwardedHost</b>	The original host requested by the client if forwarded by proxy / load balancer as seen by the SDK. <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-Host">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-Host</a>
<b>RequestHttpHeader_XForwardedProto</b>	The client's protocol (HTTP/HTTPS) if forwarded by proxy / load balancer as seen by the SDK. <a href="https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-Proto">https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Forwarded-Proto</a>
<b>MatchedConfig</b>	The name of the Action that the debug link matched
<b>TargetUrl</b>	The debug links target URL – if any.
<b>QueueConfig</b>	Information about the configuration that the debug link matched (if no match is will be empty):

	<p>EventId: matched Waiting Room Id</p> <p>Version: Configuration file version</p> <p>QueueDomain: domain of where the waiting room is hosted</p> <p>CookieDomain: the domain where the session cookie is stored once the visitor is verified.</p> <p>ExtendCookieValidity: True or False</p> <p>CookieValidityMinute: Validity time in minutes</p> <p>LayoutName: Name of the custom theme to use, if different from what is specified on the waiting room.</p> <p>Culture: Culture to use, if different from what is specified on the waiting room.</p>
--	---

*NB: At the moment the debug link is not available in the GO Queue-it Platform. Please contact support through the Support link in the GO Queue-it Platform to get a debug link.*

### 5.5.2 Error Pages

To help with the troubleshooting process, Queue-it's Error Page is designed to show specific messaging based on the nature of the error the visitor had encountered. Listed below are the different error IDs one could encounter, as well as the scenarios in which they would be presented:

Error ID	Reason	Description
1	UnknownCustomerId	A missing or invalid <b>customer ID</b> has been passed into a waiting room URL
2	UnknownEventId	<p>The waiting room is in Idle phase with the <b>Hide event</b> Idle Logic applied</p> <p>Or</p> <p>A missing or invalid <b>event ID</b> has been passed into a waiting room URL</p>

3	EventNotFound	Not in use
4	AfterWindow	<p>The visitor has exceeded the <b>Queue number validity time</b> (<b>Waiting room</b>   <b>Settings</b>   <b>Display</b> tab)</p> <p>Or</p> <p>The visitor's QueueId has been cancelled due to cancel action, cancel API call, or CAPTCHA replay</p>
5	MaxNoOfRedirectsPerQId	<p>The visitor has exceeded the <b>Max redirects per user</b> (<b>Waiting room</b>   <b>Settings</b>   <b>Display</b> tab)</p> <p>This can be exceeded because of link sharing or reuse, integration errors, or technical issues on the target website</p>
6	KnownUserV3	The <i>queueittoken</i> query string parameter is invalid due to expiry, configuration error, or tampering (section 5.4.4)
7	ChallengeSolveTimeout	The visitor took too long to solve the challenge (e.g., CAPTCHA)
8	ChallengeBlocked	The visitor's browser has blocked scripts from Queue-it's domain; challenges (e.g., CAPTCHA) cannot be presented
9	InvalidQueueitEnqueueToken	<p>For waiting rooms with <b>Require enqueue token</b> enabled, the visitor did not provide a valid token</p> <p>A token is invalid if it has been tampered with or expired</p>
10	MissingCustomDataKey	For waiting rooms with <b>Require user identification key: Require Key</b> enabled, the visitor did not provide a key
11	CustomDataUniqueKeyViolation	For waiting rooms with <b>Require user identification</b>

		<b>key: Require Unique Key</b> enabled, the visitor provided a key that is not unique
--	--	---

## 6 JavaScript Integration Method Details

JavaScript is a client-side integration where the JavaScript code is inserted on the relevant pages (preferably in the HTML <head>-section) that are to be protected by the waiting room.

### 6.1 JavaScript Integration

Once a waiting room has been created in the GO Queue-it Platform (<https://go.queue-it.net>), set JavaScript as the Integration method under the Deployment Tab in the **Manage | Waiting Rooms Settings** page. JavaScript can also be set as the default integration method in the **Account | Company Profile** page for it to be the initial setting for new waiting rooms.

Once JavaScript has been set as the integration method on a waiting room, the Integration section contains a JavaScript snippet which should be copied and inserted into in the HTML <head>-section of the relevant pages that are to be protected by the queue. Below is an example with the CustomerID: "ticketania" in the GO Queue-it Platform.

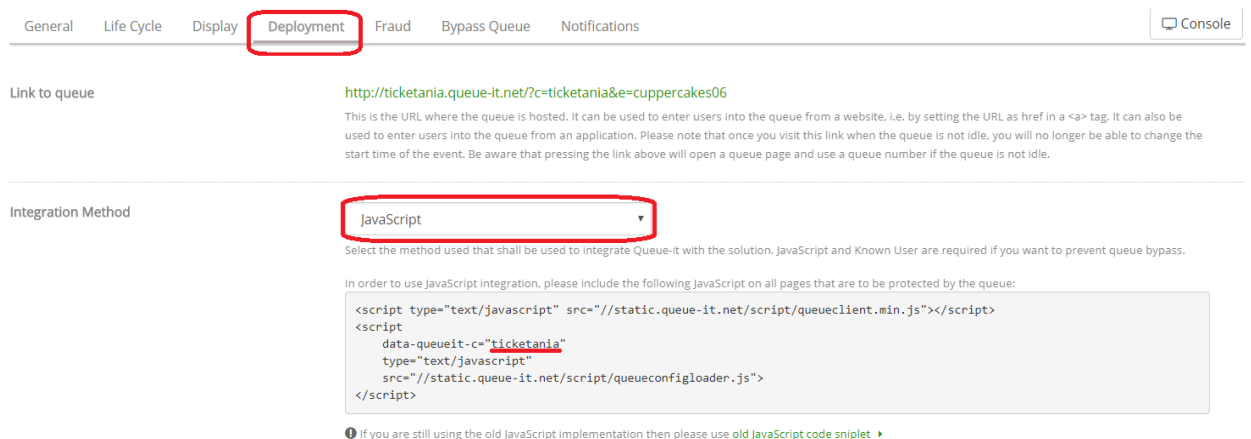


Figure 15: Event Deployment

### 6.2 JavaScript Integration Snippets

See below examples of JavaScript used in pages to protect them and redirect activity to the queue.

It is important to enter the appropriate "Customer-ID" if the text is copied from the JavaScript snippet in this white paper. The customer ID can be found in the GO Queue-it Platform under **Account | Company Profile | Customer Id**

### 6.2.1 Simple JavaScript Integration Snippet:

```
<script type="text/javascript" src="//static.queue-
it.net/script/queueclient.min.js"></script>

<script
  data-queueit-c="CustomerID"
  type="text/javascript"
  src="//static.queue-it.net/script/queueconfigloader.js">
</script>
```

### 6.2.2 JavaScript Integration Snippet for Known User AJAX Protection

Use this version of the Queue-it JavaScript ONLY if you have integrated the KnownUser server-side SDK and are looking to add protection for in-page AJAX calls

```
<script type="text/javascript" src="//static.queue-
it.net/script/queueclient.min.js"></script>

<script
  data-queueit-intercept-domain="{YOUR_CURRENT_DOMAIN}"
  data-queueit-intercept="true"
  data-queueit-c="{YOUR_CUSTOMER_ID}"
  type="text/javascript"
  src="//static.queue-it.net/script/queueconfigloader.min.js">
</script>
```

## 6.3 Benefits

The JavaScript integration method is fast and easy to implement as it is simply inserted into the HTML of a web page. It can take as little as 10 minutes to implement with JavaScript and protect a site's infrastructure from website peaks. It needs to be added to any pages of a web site that need to be protected by the queue.

It is important to note that a queue implemented with the JavaScript method can be bypassed. As JavaScript is executed client-side, it can always be manipulated by a skilled person, making it possible to bypass a queue. If it is required that visitors cannot bypass a queue, the "Known User" implementation method is advised. It is back-end code that does not execute in the client, so it cannot be avoided. See



[section 3.2.1 \(Known User\)](#) and [section 5 \(Known User Integration Method Details\)](#) for more information.

A JavaScript queue can be combined with “Known User” queues, e.g. JavaScript queues for browsing pages and “Known User” queues for cart/basket/check out.

### 6.3.1 Note: Queue-it Backend Response

The JavaScript solution will only show the waiting room if the Queue-it backend replies positively to a queuing request and if the waiting room is not disabled.

This means that the customer’s website will continue to function in any of these exceptional situations:

- Amazon AWS data centers fail (all or partially)
- The visitor’s browser client’s access to Queue-it is broken, e.g. by network failures, routing errors, firewall configurations etc.
- DDoS attacks towards Amazon AWS or Queue-it
- Queue-it malfunctions
- Other connectivity issues

In such situations, the browser will ignore the missing JavaScript call, and the website will continue as if the JavaScript did not exist. No errors will be presented to the visitor, and they will not be sent to the waiting room.

## 6.4 Configuring Waiting Rooms

The configuration of where and when the waiting room should protect the website and which Custom theme and language to use is all controlled via the Integration Configuration by specifying a set of Triggers and Actions. A Trigger is an expression matching a single, multiple or all URLs on a website. When a visitor enters the website and the URL matches a Trigger-expression the corresponding Action will be invoked. The Action specifies which queue the visitor should be sent to. In this way, changes to the behavior of the queue can be modified without the need to redeploy code to the website.

For general information about how to setup Triggers and Actions, please see [section 3.1 \(Integration Configuration\)](#).

There are however certain elements of the Queue Action that is only relevant in connection with the JavaScript integration described in this white paper. These special features are described in the next section.

### 6.4.1 On the JavaScript Tab

On the JavaScript tab it is possible to insert snippets of JavaScript that will be executed in three different scenarios:

1. OnVerified: Once the visitor has been through the queue and is verified (or if the waiting room is disabled) this script will get executed
2. OnDisabled: If the waiting room is in the Idle phase then this script will get executed
3. OnTimeout: If the call to the Queue-it back-end times out (after 5 seconds) this script will get executed

This feature can be used for many different purposes e.g. setting a cookie that can be used for later verification or to make skipping the queue a bit more difficult.

The latter is described in the next section.

## 6.5 Making Bypassing the Queue More Difficult

The JavaScript implementation obviously only works on devices / browsers where the JavaScript code is executed. There are two ways whereby the JavaScript is not executed by the browser:

1. JavaScript is disabled in the browser
2. The browser has a plug-in installed that blocks JavaScript execution

Different experiments show that the ratio of devices / browsers that do not execute JavaScript is in the 1.1% range, where (2) is the most common, probably in the 0.9% range. Most noscript type plug-in's do only block the execution of the JavaScript, but do not set the browser in a state where it will react on a <noscript> html tag.

Note: If ensuring visitors are not able to bypass the waiting room is critical for the business, please refer to the Known User Integration described in [section 3.2.1 \(Known User\)](#) and [section 5 \(Known User Integration Method Details\)](#).

The following two methods describe how to hide the content of the webpage until the visitor has been verified by Queue-it. This makes it more difficult for visitors to use with website with devices / browsers that prevent the Queue-it JavaScript from executing.

### 6.5.1 Method 1: When Using JQuery

#### 6.5.1.1 Add the below Tags in Head Section

- <noscript> tag

The <noscript> tag will redirect the device / browser automatically to a /nojavascript.htm page on the system. This page can contain info about the requirement and maybe instructions for enabling JavaScript.

```
<noscript>
<meta http-equiv="refresh" content="0; url=/nojavascript.htm">
</noscript>
```

- <div id="nojavascript">

This div will be visual as default and only hidden if the JavaScript is working.

```
<div id="nojavascript">
<script>document.getElementById("nojavascript").style.display =
"none"</script>
JavaScript not enabled. Click <a
href="/nojavascript.htm">here</a> for more information
</div>
```

#### 6.5.1.2 Add the Following Tag in Body

- <div id="page" style="display:none">

This div containing the normal page content will be non-visual as default, and only visual if the JavaScript is executing.

```
<div id="page" style="display:none">
Normal Page content goes here
</div>
```

#### 6.5.1.3 Update JavaScript Tab Settings

On the JavaScript tab in the GO Queue-it Platform, at the JavaScript Action insert this script on both the OnVerified and OnDisabled.

```
$(document).ready( function() {
$("#page").css("display", "block");
});
```

Basic
Advanced
JavaScript

OnVerified script

\$(document).ready( function() {  
\$("#page").css("display", "block");  
});

The JavaScript code specified here will be executed once the end-user has been through the queue and is verified. (do not add script tag)  
You can use `customerId` and `eventId` variable in your script as local variable.

OnDisabled script

\$(document).ready( function() {  
\$("#page").css("display", "block");  
});

The JavaScript code specified here will be executed if the queue is in a disable state. (do not add script tag)  
You can use `customerId` and `eventId` variable in your script as local variable.

OnTimeout script

OnTimeout JavaScript callback

The JavaScript code specified here will be executed if QueueId can not be verified within fixed time interval. (do not add script tag)  
You can use `customerId` and `eventId` variable in your script as local variable.

Figure 16: JavaScript Tab

## 6.5.2 Method 2: When Not Using JQuery

### 6.5.2.1 Add the Following Tags in Head Section

```

<meta charset="utf-8" />
<title>Title</title>
<noscript>
<meta http-equiv="refresh" content="0; url=/nojavascript.htm">
</noscript>

<script>
var tryEnablePage = function()
{
    var javascriptinitEL = document.getElementById("javascriptinit");
    if(javascriptinitEL)
        javascriptinitEL.style.display = "none";

    var pageEL = document.getElementById("page");
    if(pageEL)
        pageEL.style.display = "block";
}
</script>

```

### 6.5.2.2 Add the Following Tags in Body Section

```

<div id="javascriptinit"> JavaScript initializing...</div>
<div id="nojavascript">
<script>document.getElementById("nojavascript").style.display =
"none"</script>
JavaScript not enabled. Click <a href="/nojavascript.htm">here</a> for
more information
</div>
<div id="page" style="display:none">
Normal Page content goes here
</div>
<script type="text/javascript">
if(window.__queueIsInitiated)
{
    tryEnablePage();
}
</script>

```

### 6.5.2.3 Update JavaScript Tab Settings

On the JavaScript tab in the GO Queue-it Platform, at the JavaScript Action insert this script on both the OnVerified and OnDisabled.

```

window.__queueIsInitiated=true;tryEnablePage();

```

## 6.6 Queue-it Session Cookie

## 6.7 JavaScript Integration for Single Page App (SPA)

In these applications the waiting room behavior can be triggered by evaluating the value of either a cookie or a JavaScript variable or by using navigation.

To verify that the visitor still has a valid session (and extend if specified so on the **Integration | Configuration**) there is a `Queue.validateUser()`; global JavaScript function. The same function can also be used to verify if changes to a cookie or JavaScript variable should trigger the queue. It is a very lightweight function that can be called on each step of the user journey.

The `QueueIt.validateUser();` function evaluates against the latest downloaded Integration Configuration file. If you need to refresh the Integration Config file and evaluate triggers, you can call `QueueIt.validateUser(true);`.

### 6.7.1 Integration for Single Page Applications that use URL navigation

Some applications use the browser's History functions to implement routing. Examples of this include React, Angular, and Vue.

In such cases, the `data-queueit-spa` attribute has to be used. An example of this can be seen in the following snippet:

```
<script type="text/javascript" src="//static.queue-  
it.net/script/queueclient.min.js"></script>  
  
<script  
  data-queueit-spa="true"  
  data-queueit-c="{YOUR_CUSTOMER_ID}"  
  type="text/javascript"  
  src="//static.queue-it.net/script/queueconfigloader.min.js">  
</script>
```

The `data-queueit-spa` attribute enables a feature of the JavaScript integration that keeps track of navigation. All routing will be validated using the triggers and actions.

JavaScript events can then be used to get information about the visitor's waiting room journey. The example below dispatches JavaScript events when the visitor's state changes.

Basic

Advanced

JavaScript

**OnVerified script**

```
window.dispatchEvent(new CustomEvent('queuePassed'))
```

The JavaScript code specified here will be executed once the end-user has been through the queue and is verified. (do not add script tag)

You can use `customerId` and `eventId` variable in your script as local variable.

**OnDisabled script**

```
window.dispatchEvent(new CustomEvent('queueDisabled'))
```

The JavaScript code specified here will be executed if the queue is in a disable state. (do not add script tag)

You can use `customerId` and `eventId` variable in your script as local variable.

**OnTimeout script**

```
window.dispatchEvent(new CustomEvent('queueTimeout'))
```

The JavaScript code specified here will be executed if QueueId can not be verified within fixed time interval. (do not add script tag)

You can use `customerId` and `eventId` variable in your script as local variable.

These events need to be handled inside the application, depending on the user journey. They will be dispatched whenever an action matches upon navigation.



## 7 Link To Waiting Room Integration Method Details

Visitors enter the waiting room by redirecting them to [http://\[companyId\].queue-it.net](http://[companyId].queue-it.net) (e.g. if companyId=ticketania, then URL will be <http://ticketania.queue-it.net>).

This can be done by implementing the link somewhere on a website (i.e. by setting the URL as href in a <a> tag) or using a function from the application or webserver. Some parameters like c and e are required, and some are optional. It could be: <http://ticketania.queue-it.net?c=ticketania&e=test>.

Link To Waiting Room is useful in conjunction with a marketing campaign that sends an email or creates a waiting room specific landing page. By linking to the waiting room directly, the visitor is taken directly to the Queue-it infrastructure and no load is put on the target server until the visitor has completed their waiting period in the waiting room.

This link may also be shortened with the use of a URL shortener like goo.gl or bit.ly.

### 7.1 Parameters

Queue-it uses the following parameters in the URL:

- c (required): Company ID. It is used to identify the account in the <https://go.queue-it.net> administration portal, e.g. <http://ticketania.queue-it.net/?c=ticketania&e=test>
- e (required): Waiting Room ID. It is used to identify the waiting room in the <https://go.queue-it.net> administration portal, e.g. <http://ticketania.queue-it.net/?c=ticketania&e=test>
- t (optional): Target parameter. The queue now supports the capability for visitors to be redirected to multiple locations by the URL specified in the application. This allows the use of target URLs that are customized for the specific visitor, all within a single queue. All that is needed is to include the *URL encoded* target URL in a 't' parameter of the queue URL. E.g. if the intent is to redirect the visitor to [www.google.com](http://www.google.com), use: <http://ticketania.queue-it.net/?c=ticketania&e=test&t=http%3A%2F%2Fwww.google.com>
- t\_[key] (optional): This parameter allows adding data into the target URL defined in the waiting room settings. e.g. The following example defines a target URL with 2 target URL parameters ('show' and 'code'):

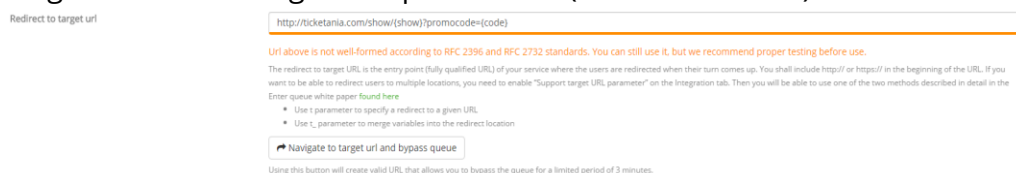


Figure 18: Redirect to Target URL

- It is possible to then add the query string parameters 't\_show' and 't\_code' from the original site which will then be combined with the target URL. Entering the waiting room with [http://ticketania.queue-it.net/?c=ticketania&e=test&t\\_show=78&t\\_code=WefjRt](http://ticketania.queue-it.net/?c=ticketania&e=test&t_show=78&t_code=WefjRt) will redirect the visitor to <http://ticketania.com/show/78?promocode=WefjRt>
- cid (optional): the cid-parameter overwrites the language / culture parameters, which are set up on the referenced event to be used on "Before", "Queue", "After", "Exit" and "Error" pages. If the cid-parameter is invalid or not found the pages will be displayed in the waiting room language / culture. The cid-parameter shall specify a valid CultureID e.g. <http://ticketania.queue-it.net/?c=ticketania&e=test&cid=da-dk> (this will overwrite language to da-DK / Danish). Alternatively, the waiting room could be set to "Detect language from browser settings"
- l (optional): the l-parameter overwrites the theme that is set up on the referenced waiting room to be used on "Before", "Queue", "Error" and "After" pages. If the l-parameter is invalid or not found the pages will be displayed with the waiting room theme, e.g. <http://ticketania.queue-it.net/?c=ticketania&e=test&l=test> (will overwrite theme to "test")

## 8 Queue Skipping

There are two ways to skip(bypass) the queue, either by specific IP address or via Link Skipping.

### 8.1 Skip by IP Addresses Functionality

Queue-it “Queue Bypass by IP Addresses” is a feature that can be added to Pro and Enterprise subscriptions.

Queue Bypass allows specific IP addresses to bypass the waiting room. For example, it can be used for call centers, where there is a need for employees to interact with the protected website without being queued.

Visitors entering a waiting room will be handled as if the waiting room was disabled, i.e. redirected with:

- QueueId = 00000000-0000-0000-0000-000000000000
- Redirect type =Disabled
- a valid hash-parameter

The bypass is inserted after Queue-it DDoS and scripting protection.

### 8.2 IP Address Groups

IP Address Groups can be maintained from **Manage | IP Address Groups** in the GO Queue-it Platform. Once IP address groups are created, they can be enabled / disabled on each waiting room. (**Waiting room | Settings | Bypass Queue** tab), specifying a collection of IP addresses to bypass the waiting room, rather one by one.

### 8.3 Bypass Link

Bypass link is a direct link to the target website. It can be generated for a waiting room (**Waiting room | Settings | General** tab) and allows the administrator to check the performance of the website without viewing the waiting room page. To prevent it from accidentally being used to allow general traffic past the queue, each instance of this link is only valid for 3 minutes.

### 8.4 Logging

Queue-it “Notifications & Logs” is a feature that can be added to Pro and Enterprise subscriptions.

Each time a Bypass link is generated and each time an IP Address Groups is changed it is logged. With the “Notifications & Logs” feature it’s possible to find all

these changes from **Account | Change Log** menu. This gives full traceability into the bypassing of a waiting room.

## 9 Bot Management

Queue-it is not a bot protection tool but has mechanisms to prevent bots from interfering with the waiting room and from gaming queue fairness.

Queue-it integrates with Professional Bot protection tools like Distil Networks, Akamai, PerimeterX and Incapsula.

### 9.1 Bot Categories and Types

Bots have become a huge topic on the internet and can be understood by category:

- Crawler: A Web crawler, sometimes called a spider, is an Internet bot that systematically browses the Web, typically for indexing
- Feed Fetcher: Scrapes content from sites and republishes it
- Scraping: Data scraping used for extracting data from websites
- Search Engine: Indexes the internet
- Service Agent: Purpose-built software systems that run a specific set of functionalities on a site
- Site Monitor: Used by businesses to ensure website uptime, performance, and functionality is as expected
- Vulnerability Scanner: Computer program designed to assess computers, computer systems, networks or applications for weaknesses

#### 9.1.1 Bot Types (Good vs. Bad Bots)

Bots / scripting can be grouped as follows

- Good bots that you want to have on your site
- Bad bots that interfere with your technical and business goals

#### 9.1.2 Good Bots

Examples of good bots are:

- Search engine crawlers from e.g. Google, Bing, Yahoo and Baidu
- Advertising bots verifying site content from e.g. Google
- Content fetch bots to preview content from e.g. Facebook, Twitter, LinkedIn
- Website monitoring tools like Pingdom, Gomez, New Relic and CA App Synthetic Monitor
- Any custom-built functionalities that hit your site

Website owners would normally like to allow good bots on their site and under controlled conditions.

A special case of good bots is load test scripts. This is described in our “Queue-it Load Test white paper”.

### 9.1.3 Bad Bots

Examples of bad bots are:

- Account Takeovers
- Transaction Fraud
- Reconnaissance Attacks
- Content Theft
- Digital Ad Fraud
- Form Spam
- Price Scraping
- Slowdowns and Downtime
- Skewed Analytics
- Add-to-Cart bots

## 9.2 Bot Support in Queue-it

Queue-it has the following support for bots.

### 9.2.1 Robots.txt

The robots exclusion standard, also known as the robots exclusion protocol or simply robots.txt, is a standard used by websites to communicate with web crawlers and other web robots.

Queue-it visitor facing queue-pages have the following robots.txt file content:

```
User-agent: *  
Allow: /about  
Disallow: /
```

Queue-pages all have a unique queueids and are dynamic / individual. Queue-pages should never be hit by bots; hence the **Disallow: /**.

The reality is that most of the bad bots completely ignore this setting. Good bots like Google Search Engine crawlers support the setting and will ignore indexing queue-pages. This is of course an issue if the crawler ends up on a queue-page during the crawling of your website. See description on how to let crawlers bypass below in [section 9.2.3 \(Let your good bots bypass Queue-it\)](#).

## 9.2.2 Request Analysis

Queue-it monitors all traffic into the waiting room. If it sees suspicious traffic from an IP address (which is often initiated by automatic scripts or robots) Queue-it will block such traffic as follows:

- Set a 40x response to HTTP request at high rates
- Soft-block the HTTP requests into queue at lower rates but with an access pattern is that are different from normal human interaction with the web (e.g. very short latency and high rate of attempts)

### 9.2.2.1 Soft-blocks

A soft-blocked IP will be soft-blocked for one hour. Requests from a soft-blocked IP address will get a 302 redirect into a static page with a CAPTCHA. If the CAPTCHA is solved, the specific visitors / browser will get access to the queue-page with one queueid.

English

QUEUE-IT

reCAPTCHA™

Type the text

Privacy & Terms

You are about to be assigned a place in line. The reason for this is either because the website you want to access is not yet open for visitors, or because the website has a large number of users all wanting access at the same time. Please enter the letters you see in the box above and hit Enter to proceed.  
Please observe that this function requires Cookies to be enabled.

**Figure 19: CAPTCHA Form**

The account administrators will receive an email when soft-blocks happen. See “Queue-it Notifications and Logs white paper”.

### 9.2.2.2 Request Analysis Monitoring

An overview of the soft-blocks made by Queue-it is available in the /Statistics/Request Analysis menu. There is an overview and a detail log tab.

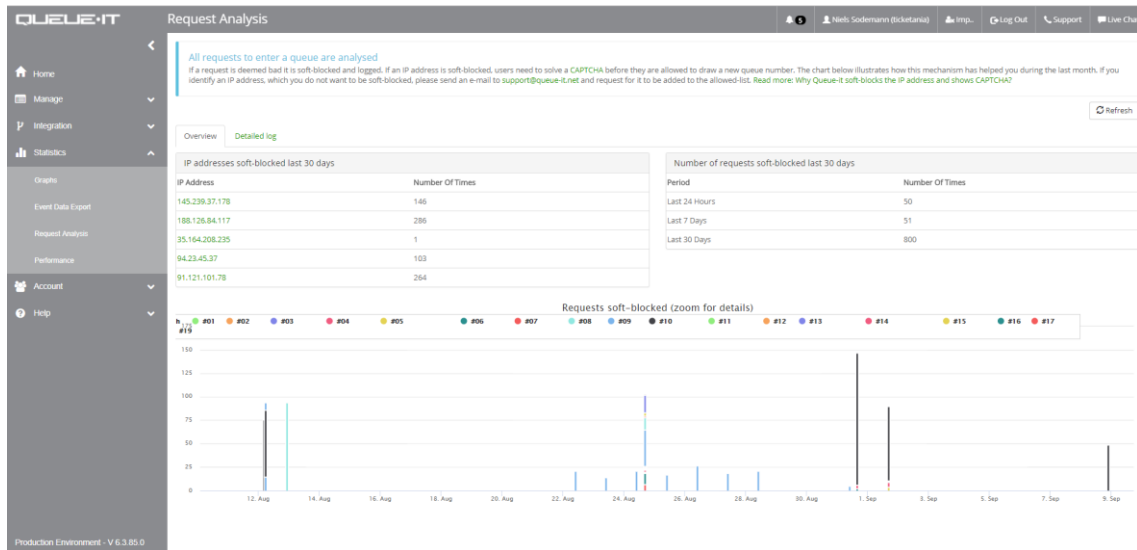


Figure 20: Request Analysis

You can click the blocked IP address to learn more about it.

### 9.2.3 Let your good bots bypass Queue-it

Good bots can be set up to be ignored by Queue-it as follows:

- Using triggers with a userAgent pattern to ignore queue actions
- Using triggers with header pattern from an existing professional bot protection tools to ignore queue actions

#### 9.2.3.1 userAgent Patterns

The userAgent attribute is set on the client side by the bot / browser and can of course be manipulated / spoofed.

Good robots e.g. come with userAgents as listed in the table below:

userAgent	Suggested Trigger Pattern
Mozilla/5.0+ (compatible; +bingbot/2.0;++http://www.bing.com/bingbot.htm)	bingbot
AdsBot-Google+ (+http://www.google.com/adsbot.html)	adsbot-google
Mozilla/5.0+ (compatible; +Googlebot/2.1;++http://www.google.com/bot.html)	googlebot
Mozilla/5.0+ (compatible; +Baiduspider/2.0;++http://www.baidu.com/search/spider.html)	baiduspider
Feedfetcher-Google; (+http //www.google.com/feedfetcher.html)	FeedFetcher-Google



Google-Adwords-Instant (+http://www.google.com/adsbot.html)	Google-Adwords-Instant
Facebot	facebot
facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php)	facebookexternalhit
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/534+ (KHTML, like Gecko) BingPreview/1.0b	BingPreview

Business usually want good bots to be allowed in on all descriptive catalog pages – but not even a good bot like Googlebot should be allowed in on the checkout flow, payment steps and other personalized pages.

In the example below, the trigger “Ignore good bots” would trigger for the userAgents in the table above, but only on pages that does not contain /checkout:

**Queue-it Trigger** edit existing

Name \*

A descriptive name of the Trigger

Condition expressions for which match will be evaluated: 'Type' specifies a category of validator. 'Parameter' specifies an evaluation criteria to be matched. 'Value' takes a string input for an expression

AND	Type	Parameter	Operator	Value	Ignore Case
<input checked="" type="checkbox"/>	UserAgent		ContainsAny	baiduspider, adbot-google, googlebot, bin...	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	Url	PageUrl	NotContains	/checkout	<input checked="" type="checkbox"/>

[Return To Trigger List](#) [Save Changes](#)

**Figure 21: Exclude bots Trigger**

The above Trigger must then be used to trigger an Ignore Action as shown below (observe that the order of Actions matter, so the Ignore Action must be first in the list)

**Queue-it Actions**

**Warning:** You have unpublished changes  
Some of your action(s), trigger(s) or configuration(s) have been updated. Please be aware that until you publish, these changes will have no affect, e.g. your previous published integration version will be still be used.  
To perform a publish please visit integration overview page.

[Save Ordering](#) [Add New Action](#)

Action Type	Name	Event	Configuration	Triggers	Integration Type
1 <input checked="" type="checkbox"/> Ignore	Ignore Good Bots	N/A	N/A	Ignore good bots	KnownUser
2 <input checked="" type="checkbox"/> Queue	Ticketania	zara20180709	Default (Queue-it)	Ticketania domain	KnownUser

[Delete](#) [Edit](#) [Reorder](#)

The userAgent functionality trigger works with both JavaScript (front-end) and “Known User” (server-side) Queue-it integrations.

### 9.2.3.2 Header Patterns

Professional bot protection tools like Distill Networks, Akamai, PerimeterX and Incapsula can set request headers based on the setup / findings on the individual request.

For example, Akamai can add a header named Akamai-Bot to each request forwarded to the origin server. This header identifies what type of bot it is (Customer-Categorized, Akamai-Categorized, or Unknown) and includes Bot ID, response action, and category/detection method. The format is:

```
BOT TYPE (BOTNET_ID) :ACTION:BOT_CATEGORY.
```

As an example, the following request header would be added for Googlebot requests if Search Engine Bots were in Akamai "Monitor":

*Akamai-Bot:Akamai-Categorized Bot (googlebot):Monitor*

To avoid queueing these bots,, simply add a header trigger, with parameter = Akamai-Bot and value Contains (ignore case) = bot and link use it to an ignore action as shown below:

Name \*   
A descriptive name of the Trigger

Condition expressions for which match will be evaluated: 'Type' specifies a category of validator. 'Parameter' specifies an evaluation criteria to be matched. 'Value' takes a string input for an expression

AND	Type	Parameter	Operator	Value	Ignore Case
	HTTPHeader	Akamai-Bot	Contains	bot	<input checked="" type="checkbox"/>

[+](#)

Action Type	Name	Event	Configuration	Triggers	Integration Type
1  Ignore	Allow Good Bots to Bypass the Queue	N/A	N/A	<a href="#">Good Bots - Akamai</a> ⓘ	KnownUser

**Figure 22: Akamai-Bot**

The ignore action should be above all Queue actions because the evaluation is done top down.

The Http Header trigger type can only be used with "Known User" (server-side) Queue-it integration.

## 9.2.4 Proof-of-Work Challenge

Queue-it's Proof-of-Work challenge uses computing processing power as the cost driver to block bad bots. The more queue numbers that bad actors seek to obtain, the more CPU is drained and the more costly their project becomes.

Proof-of-Work is configured under Bots and Abuse tab on waiting room settings. Refer to Queue-it Bots and Abuse Management White Paper for details.

## 10 Visibility Modes

The Visibility modes are controlled from **Waiting Room Setting | Display tab**.

Queue-it supports the following Visibility modes:

- Visible At Peak
- Always Visible

### 10.1 Visible At Peak

When 'Visible At Peak' is selected and visitor traffic accessing the target site is higher than the max redirects allowed per minute, then the waiting room will be shown to visitors before they access the target site. If visitor traffic does not exceed the max outflow allowed per minute, visitors will proceed directly to the target site.

Visible At Peak works by combining visual queue-mode for some visitors with non-visual queue-mode for others to set the total outflow as close to the set threshold as possible, thereby optimizing the outflow in that situation.

#### 10.1.1 Optimization Options for Visible At Peak Mode.

By default, waiting rooms are Outflow Optimized and options are not visible in the GO Queue-it Platform. In rare cases, waiting rooms need to be **Burst Optimized** rather than the default, **Outflow Optimized**. This option can be enabled in the GO Queue-it Platform on a case-by-case basis. Consult Queue-it if a high burst in inflow (like greater than ten times Max Outflow) in less than 3 minutes is witnessed.

Irrespective of waiting room options, the first 2 seconds of any inflow will be redirected as non-visible redirects. Using Burst Optimized Mode will create false positives and should only be used in special situations where the inflow spikes are not generated by natural visitor inflow arriving at the protected site.

Using Visible At Peak, Burst Optimized Mode, the queue mode will kick in if the accumulated inflow over 2 seconds exceeds approximately 20% of Max Outflow pr minute.

Example:

Using Burst Optimized Mode that has Max Outflow pr minute value of 1200, the queue will kick in if approximately 240 visitors arrive in 2 seconds.

By comparison, using Outflow Optimized Mode, the queue will not kick in instantaneously, avoiding false positives when inflow is irregular.

## **10.2 Always Visible**

When 'Always Visible' is selected, the waiting room will be visible to all visitors before they access the target site. By default, waiting rooms are Outflow Optimized.

## 11 Custom Theme

The Custom Theme feature allows for the modification of the visual representation of the visitor facing pages of the queue system. This can be done to match the brand or aesthetic of a website. Please refer to the “Custom Theme White Paper” located here: <https://go.queue-it.net/help/resources>

### 11.1 Default Theme

Queue-it has a Default Theme which is available to all our customers when customers do not buy the Custom Theme feature. With the Default Theme the customer still can insert their own logo for branding their waiting room.

### 11.2 Customized Theme

With the Custom Theme feature it is possible to change the visual representation of the pages involving the queue (Before page, Queue page, etc.) to match the design of a waiting room. Examples of customized themes from a variety of customers are located here: <https://queue-it.com/custom-queue-pages/>

### 11.3 Language and Culture

Queue-it currently supports 43 languages. Language setting is configured in the waiting room Display tab of the GO Queue-it Platform

Additional languages can be added if there is a compelling business requirement.

### 11.4 Multilingual Sites

The best way of handling multilingual sites is to use Actions and Triggers to have the waiting room page language follow the language on the website. The language of the site is normally found in the URL (e.g. /en and /fr urls), cookies or JavaScript parameters.

You would create two Triggers, one for each language and two actions, one action for the least used language and a default action. On the first Action use the less used language and force the language on that action. The second will just be default.

The “Detect language from browser settings” option in the Display tab of the Waiting Room should only be used if this is not possible.

## 11.5 FQDN - (Using another FQDN for branding purpose)

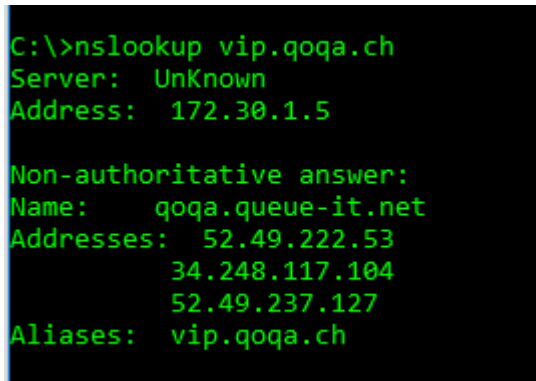
This feature is available on Pro and Enterprise subscriptions.

It is possible to use another fully qualified domain name (FQDN) rather than [http://\[customerId\].queue-it.net](http://[customerId].queue-it.net) to align the visitor experience to the company's brand, e.g.: <http://queue.ticketania.com>.

This is done by creating a CNAME record in the company's DNS pointing to [\[customerId\].queue-it.net](http://[customerId].queue-it.net). The format for this is typically like "waitingroom.ticketania.com CNAME ticketania.queue-it.net".

Note: the https (ssl/433) option is not directly supported when using CNAME since a SSL certificate is needed for the sub domain to support that

For Pro-subscriptions, a special agreement can be made with Queue-it for supporting the https option. Send the preferred FQDN to [support@queue-it.com](mailto:support@queue-it.com) with a request for this feature. Assure that the CNAME record is in place before making this request. You can verify it is working by using nslookup. You should see something like this:



```
C:\>nslookup vip.qoqa.ch
Server: UnKnown
Address: 172.30.1.5

Non-authoritative answer:
Name: qoqa.queue-it.net
Addresses: 52.49.222.53
           34.248.117.104
           52.49.237.127
Aliases: vip.qoqa.ch
```

Figure 23: nslookup Example

Once the CNAME is in effect and the new domain Queue-it SSL certificate is created (and Queue-it has sent notification of this) you may place the new FQDN in the Domain alias (cname) entry on the Deployment page of your event via the Go Platform. You may also add it as a Default domain alias in your "Company Profile" settings via the "Account" menu in the GO Queue-it Platform.

## 12 Widgets

Widgets allow the ability to add inline queue functionality to a website using HTML and JavaScript.

This is a powerful way to handle massive opening scenarios where an initial static HTML page (e.g. served by the load balancer as a “maintenance” page) is used to offload visitors directly in a controlled way without exposing info about where the queue is hosted until the pre-queue phase starts.

Widgets are generated by JavaScript and the following reference must be added once per webpage that includes widgets. The reference can be put anywhere in the HTML document.

```
<script type="text/javascript" src="https://static.queue-it.net/script/queueit.min.js"></script>
```

### 12.1 Enter Queue Button

The **Enter Queue** button widget allows a button to be easily inserted on a website that will send the visitor to the queue.

The button supports the waiting room lifecycle with 4 states (idle, pre-queue, queue and post-queue) that will automatically change the button behavior matching the different phases. To insert the button, insert the following JavaScript/HTML code:

```
<span
  data-queueit-widget="Button"
  data-queueit-eventid="test"
  data-queueit-customerid="ticketania">
  <span class="queueit-noscript">Please enable javascript to show
  this content</span>
</span>
```

#### 12.1.1 Text

The text on the button may be changed to the preferred wording and language by adding the “data-queueit-texts” attribute to the element:

```
data-queueit-texts=" button: { id:'Please Wait', prequeue:'Join
Queue', queue:'Enter Queue', postqueue:'Queue Ended'}"
```

#### 12.1.2 Styling

Styling may be applied to the button using the “data-queueit-style” attribute. The value of the attribute is CSS and any style defined will only be applied to the widget.

```
data-queueit-style="button { color: blue; }"
```

### 12.1.3 Target URL

To specify a target URL (overriding the target URL defined for the Waiting room), add the “data-queueit-targetUrl” attribute to the element:

```
data-queueit-targetUrl="http://ticketania.com"
```

### 12.1.4 Target URL Parameter

Hiding information about URLs on a website, such as specific ticketing URLs or an online retail campaign URL, can be important to prevent people who want to game the system. “Target URL Parameters” can be used to merge the Queue-it target URL with client-side parameters, like:

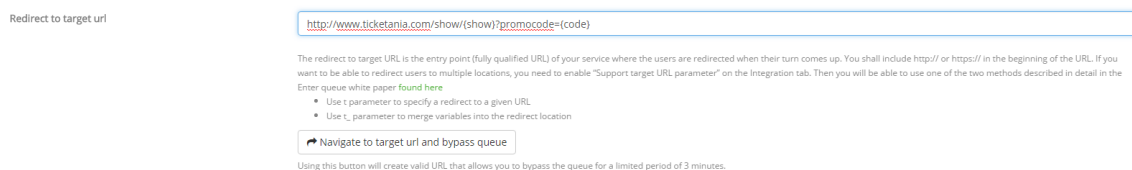
- A promotional code
- Google UTM parameters
- Etc.

To add target URL parameters, add the “data-queueit-targetUrlParams” attribute to the element:

```
data-queueit-targetUrlParams="{ show: '78', code: getCode(widget, key); }"
```

Values may be added statically or as a reference to a JavaScript function (getCode in the example above is a JavaScript code snippet returning the text “WefjRt”)

Assuming the data-queueit-targetUrlParams above and this event’s value for “Redirect to target URL”



**Figure 24: Redirect to Target URL**

The visitor would be redirected to the following URL after clicking the Button / waiting in queue:

<http://ticketania.com/show/78?promocode=WefjRt>

## 12.2 Countdown

The countdown widget will display the relative time to the waiting room start

```
Starting in 1 hour 36 minutes
<span data-queueit-widget="CountDown"
  data-queueit-eventid="test"
```



```
data-queueit-customerid="ticketania">
<span class="queueit-noscript">Please enable javascript to show this
con-tent</span>
</span>
```

### 12.2.1 Text

The text of the countdown may be changed to preferred wording and language by adding the “data-queueit-texts” attribute to the element:

```
data-queueit-texts="label: { startingin: 'Starting in',
lessthanminute: 'less than a minute', eventstarted: 'Queue has started',
eventended: 'Queue has ended', day: 'day', days: 'days', hour: 'hour',
hours: 'hours', minute: 'minute', minutes: 'minutes' }"
```

### 12.2.2 Styling

Styling may be applied to the countdown label using the “data-queueit-style” attribute. The value of the attribute is CSS and any style defined will only be applied to the widget.

```
data-queueit-style="span { color: blue; }"
```

## 13 Queue Outflow

Queue-it uses a flow-based approach and is designed so outflow (blue) is at or below the set max outflow speed in any 60 second interval. In the image below, it is shown that the 60 second interval only redirect 60 visitors (the Max outflow).

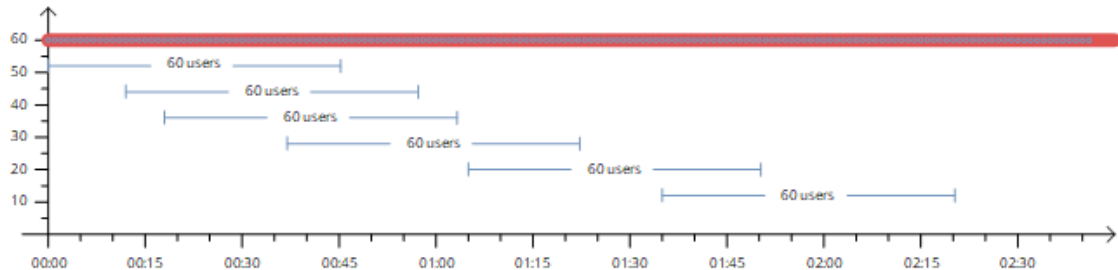


Figure 25: Visitor Redirect Interval Approach

This is managed by backend logic that keeps track of:

- Current outflow rate
- Current “no-show rate” (rate of visitors opting out of the queue, either by explicitly clicking the ‘exit line’ link or without any open browser tabs on the queue-page)
- Current “Re-entry rate” (rate of visitors re-entering the queue). “Re-entry rate” is normally only a function of technical issues

The values are sampled over a 3-minute period in the Queue-it back-end, to avoid large fluctuations.

### 13.1 The Actual Redirect

When the visitor is on the queue-page an AJAX call, `getStatus()`, gets the status of the `queueId` assigned to the visitor. This is called at a rate between 2 and 40 seconds depending on the `queueId`’s current waiting time.

The `getStatus()` call returns information about the wait etc., and whether the visitor turn is up or not. The response in `forecastStatus` = “InLine” if the visitor is still waiting.

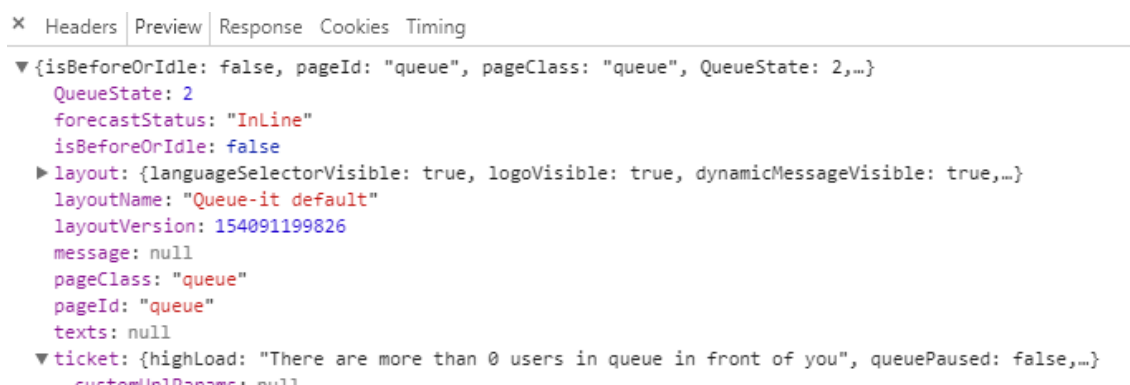
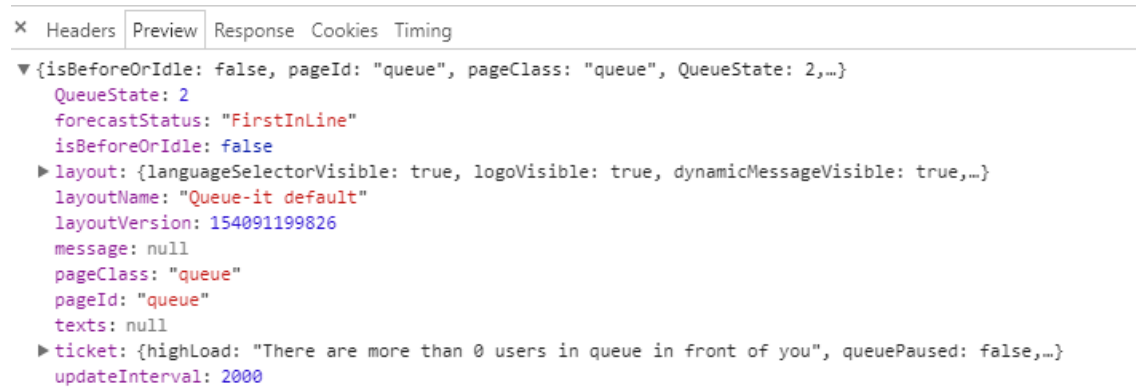


Figure 26: AJAX Responses

The `getStatus()` will eventually return `forecastStatus` is "FirstInLine". Normally the browser will make 3-5 `getStatus()` calls with 2 second intervals before the actual redirect happens. This leads to an approximately wait time in the "FirstInLine" status between 6 and 10 seconds.



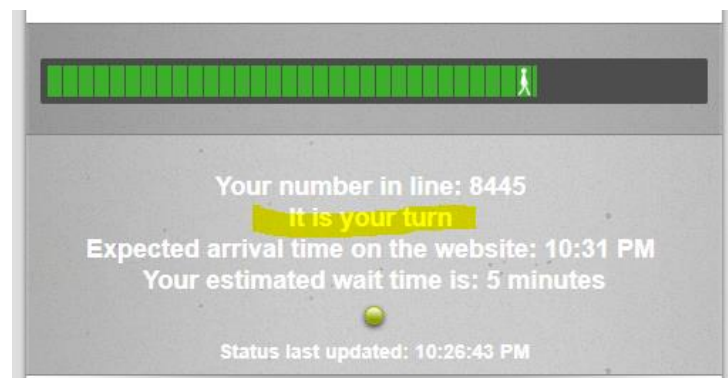
```

{isBeforeOrIdle: false, pageId: "queue", pageClass: "queue", QueueState: 2,...}
  QueueState: 2
  forecastStatus: "FirstInLine"
  isBeforeOrIdle: false
  layout: {languageSelectorVisible: true, logoVisible: true, dynamicMessageVisible: true,...}
  layoutName: "Queue-it default"
  layoutVersion: 154091199826
  message: null
  pageClass: "queue"
  pageId: "queue"
  texts: null
  ticket: {highLoad: "There are more than 0 users in queue in front of you", queuePaused: false,...}
  updateInterval: 2000

```

**Figure 27: AJAX Responses**

The visitor will see the "It is your turn" message on the queue-page.



**Figure 28: Your Turn Prompt**

And they will receive the actual signal to make the redirect, including the redirect location and the unique `queueittoken` with the hash in the URL. The token is generated server-side, preventing modification to this part.



```

{isRedirectToTarget: true, redirectUrl: "https://secure.ticketania.com/?queueittoken=e_beforettest08~q_6c1378db-5bb2-47c6-t"}

```

**Figure 29: Redirect to Target AJAX Response**

The visitor will see the "Thank you for waiting. You are now being redirected to the website" on the queue-page.

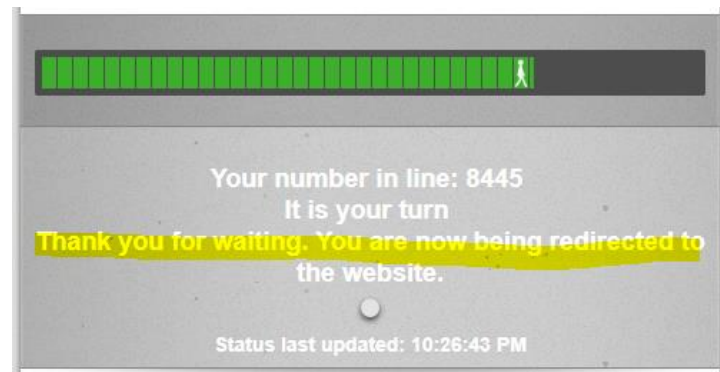


Figure 30: Now Redirecting Prompt

## 13.2 Defining Capacity and the Flow-based Approach

### 13.2.1 Traditional Way of Defining Capacity

The traditional way of defining the capacity of a web-based system is by “number of concurrent users”. This definition is widely used and is also found in Google Analytics within “active users on site” under Real-Time/Overview.

This way of defining capacity assumes that visitors are spread throughout the entire user journey. However, it does not take users entering and exiting the user-journey into account. See [section 13.2.3 \(Set Release Time\)](#), where an example of a timed-release situation (i.e. a release at 10:00 on a specific date) is shown that describes the traditional way of defining capacity and how it leads to an incorrect understanding and conclusion about capacity.

### 13.2.2 The Flow-based Approach

Our approach to considering capacity is a flow-based approach based on Little’s law. At optimum over a longer period, the throughput per time unit in a given transactional system must have an average inflow equaling the average outflow. If the average inflow is below the average outflow, the system will be emptied of users. If the inflow exceeds the outflow, the system will become congested.

Little’s law states that “*The long-term average number of customers in a stable system  $L$  is equal to the long-term average effective arrival rate,  $\lambda$ , multiplied by the average time a customer spends in the system,  $W$ ; or expressed algebraically:  $L = \lambda W$ .*” For example, if the maximum number of concurrent sessions is  $L = 1,000$  and the average session length is  $W = 10$  minutes, then the arrival rate must be:  $\lambda = L/W = 1,000 / 10$  users per minute = 100 users per minute, for the system to be stable.

The rate can be calculated using the Little’s law equation, given the maximum number of concurrent sessions, the average length of the session and a large enough sample size.

In the following example, the average inflow is 1 per time unit and each user stays in the system for an average of 20 time units.

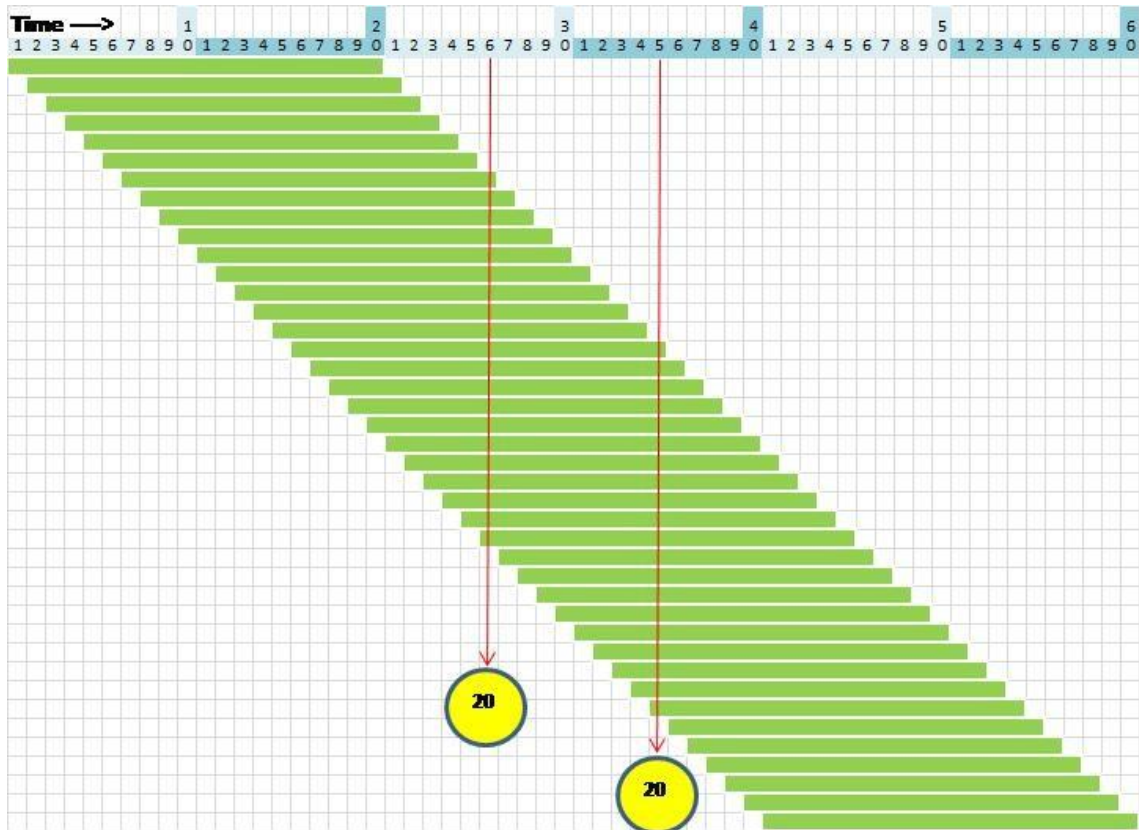


Figure 31: User Time Units

At optimum, the average amount of users in the transactional system is: average inflow x average time = 20.

### 13.2.3 Set Release Time

Within the ticketing industry, it is common that tickets go on sale at a specific and preannounced time, like 10:00 am. Increasingly, online retailers are adopting this approach, for example with Black Friday/Cyber Monday capped campaigns and launches like the Apple iPhone 00:01 release.

This tends to set an unnatural surge in website traffic and interfere with normal traffic flow. Visitors cannot continue their user journey until the given release time comes up. Even displaying the “System under maintenance” page will create the same situation with an unnatural block in flow.

Although this does not seem like a major issue, it radically changes the entire website dynamic. This is demonstrated in the following example:

Assume that:

- Website capacity is 1,000 concurrent users
- Campaign release begins at 10:00
- There is a 10 min. user-journey

So, if you have 1,000 users on the website at 10:00 and you use the flow-based approach (using Little’s law), you will get:

- Capacity to 1,000 users / 10 min. = 100 users per minute
- All 1000 users begin user-journey at 10:00

Result: overshooting capacity by 10x the first minute, as you have 1,000 users the first minute.

Furthermore, the 10 min. wait time and subsequent inflow time allotted to the 1,000 users prior to the sale will be approximately 60 min. of wait time and inflow, as visitors tend to queue early when waiting for a popular item, like an iPhone. As a result, there will be approximately 6,000 users at 10:00, as users arriving within the 60 minutes leading up to the timed release will end up waiting until the user-journey can be continued.

Therefore, capacity will be overshoot, with: 6,000 users / 100 users per minute / 1 min = 60x, in the first minute.

### 13.3 The Estimated Wait Time Calculation

The wait time is individually calculated for each visitor, just like in a physical queue. Consider the wait at the check-in counter in the airport. Individual wait time is a function of the number of people ahead, how many are serviced at the same time, how many leave the queue before their turn comes, etc.

The Queue-it visitor's current estimated wait time calculation is based on an intelligent, complex algorithm using the following inputs:

- **noua**: "Number of users ahead"
- **cmre**: Current "Max outflow" in users per minute
- **nsr**: "No-show ratio" (ratio of users opting out of the queue, sampled over the last 3 minutes). "No-show ratio" is normally a function of current wait time and time of the day
- **rer**: "Re-entry rate" (ratio of users re-entering the queue over the last 3 minutes). "Re-entry rate" is normally only a function of technical issues

For a given update, the user's expected wait time is calculated as follows:

$$\frac{noua}{cmre} * \frac{1 - nsr}{1 - rer}$$

Figure 32: Estimated Wait Time Calculation

Example:

- number of users ahead = 2,000
- current maximum redirects = 450 users per minute
- no-show ratio = 50%

- re-entry rate = 10%

Expected Wait Time = (2,000 users / 450 users per minute) \* [(1-50%) / (1-10%)] = ~ 2 minutes and 28 seconds

## 13.4 Understanding Outflow in Different Scenarios

### 13.4.1 Opening Scenario

When a waiting room has a set start time, the outflow is managed the following way:

- Until T + 00:00:15: The queue is still in pre-queue. All users arriving before this time are held in the Queue-it infrastructure but not ordered
- T + 00:00:15 to 00:00:30: Queue-it randomizes visitors arriving before 00:00:15, including the collection of pre-queued users. The system then assigns a sequential integer number in line to all
- After 00:00:30: normal first-in-first-out logic, where additional users are queued in the order at which they arrive

To ensure that any 60 second period is at / below the set threshold, the first 60 seconds are special as no outflow happens for the first 30 seconds. After which the system will send the max amount in the final 30 seconds.

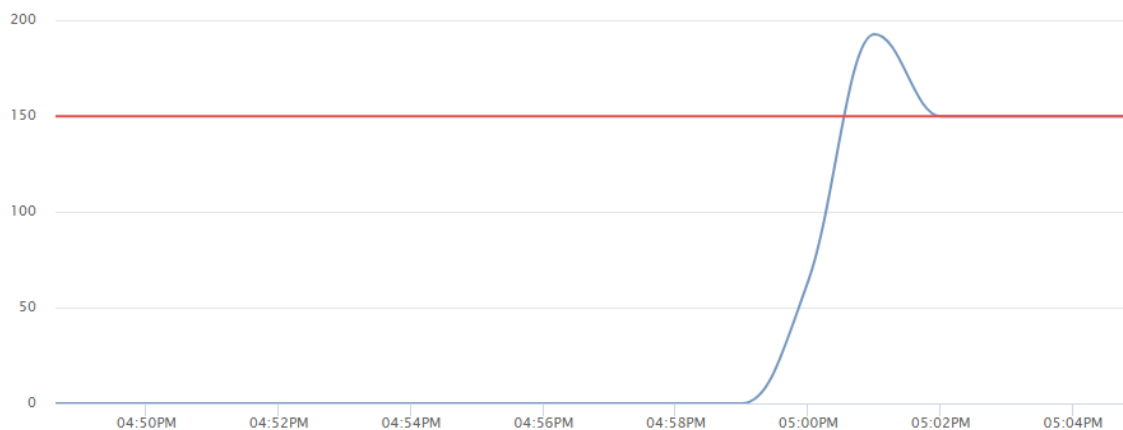


Figure 33: First Minute Redirects

### 13.4.2 Steady Flow

In the steady state and with a redirect speed > 30 user per minute, the outflow (blue) will be exactly on the set Max outflow (red).



Figure 34: Steady Flow

### 13.4.3 Steady Flow at Very Low Ratio

At lower (<20 visitors per minute) Max outflow (red), the actual outflow (blue) can vary wildly. This is simple since Queue-it only redirects an integer number of visitors (1, 2, 3 etc.) per minute and that the 60 second period described in [section 13 \(Queue Outflow\)](#) can't be divided into an integer numbers below 20 to 30 visitors per minute. For example, at 5 visitors per minute, to would lead to 60 seconds / 5 visitors per minute, giving one visitor each 12 seconds, so if one redirect just slides 1 second out of each end of the 60 second interval, you could see 7 redirects in one interval and 3 in the next. This will lead to a +/- 40% fluctuation.

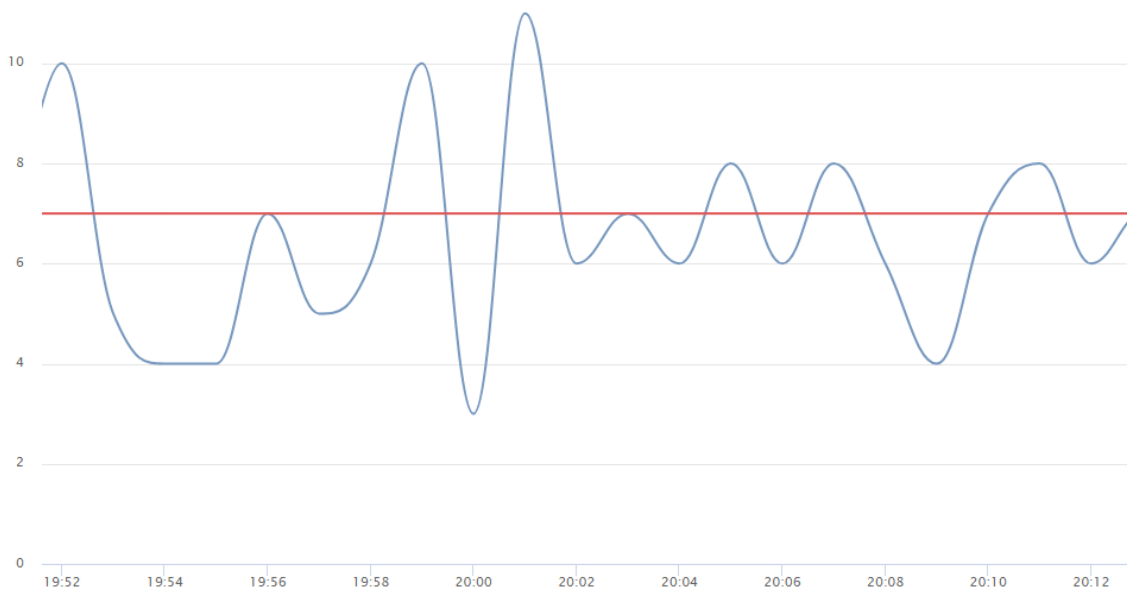


Figure 35: Steady Flow at Low Ratio



### 13.4.4 Sudden Drop in Short Period

The constant outflow requires that there is an active browser trying to be redirected. In the curve below there is a short period where the outflow drops until Queue-it throttling recalibrates. The most likely explanation for the below is an interval with robots that cannot do the redirect. It could also happen when Queue-it servers recycle, approximately one time per 24 hours.

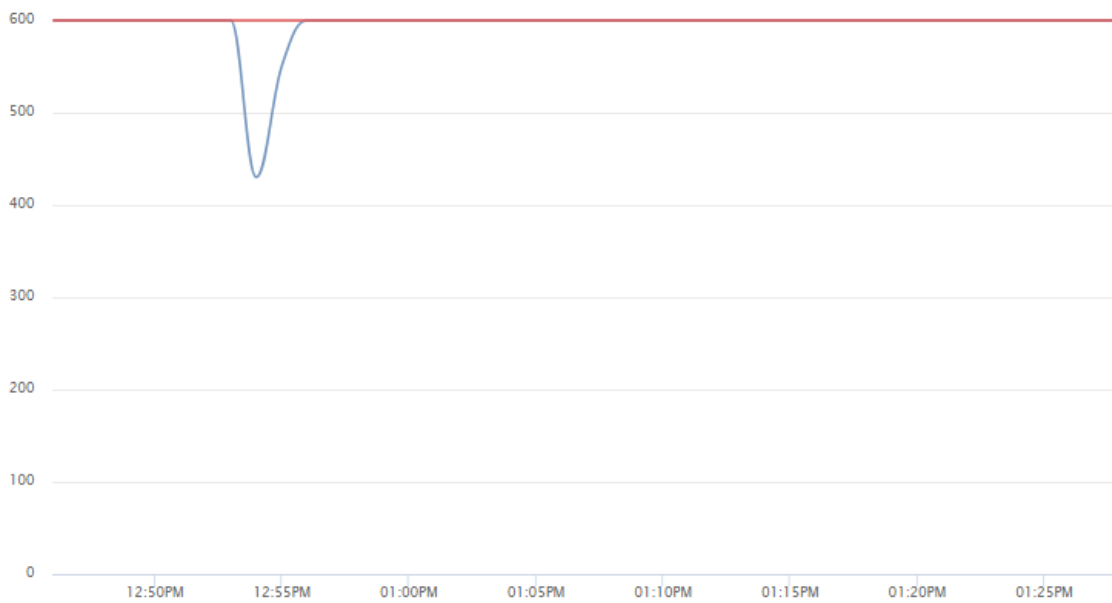


Figure 36: Sudden Drop in Short Period

### 13.4.5 Increasing Max Outflow

When the Max outflow (red) is increased, the outflow is increased with the all 60 second period has same outflow principle described in [section 13 \(Queue Outflow\)](#). This will give approx. 60 second interval from the speed increase and until outflow is at the new level.

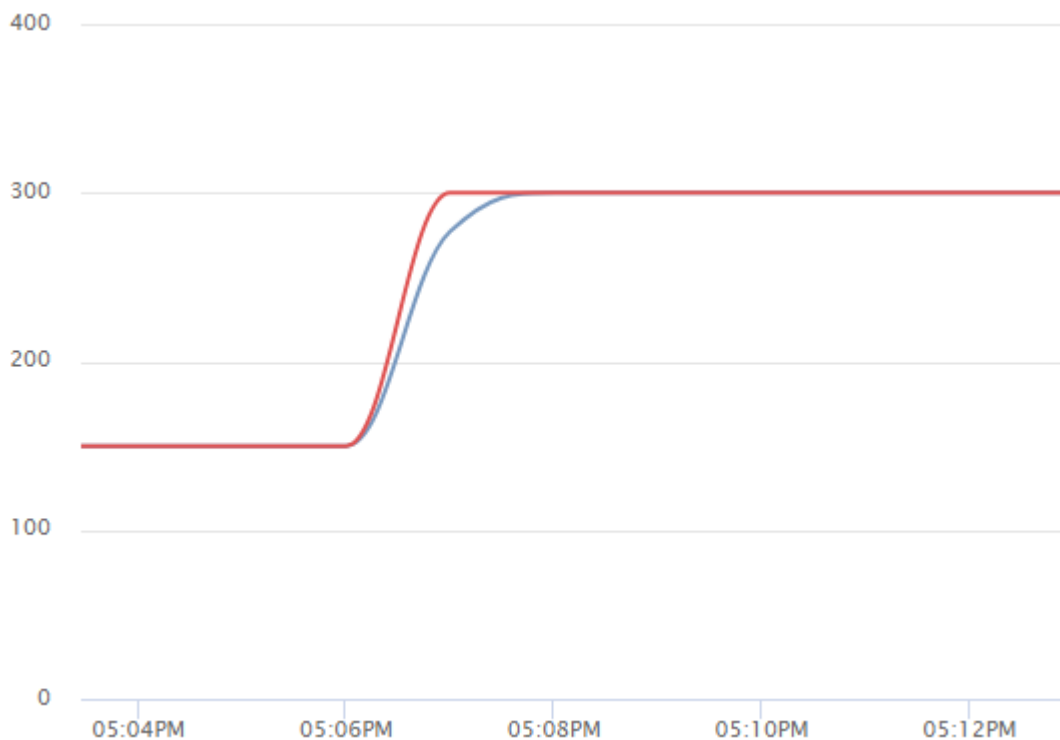


Figure 37: Increasing Max Outflow

### 13.4.6 Decrease Max Outflow

When the speed is decreased, the change happens immediately. Furthermore, the number of browsers with the forecastStatus value of “FirstInLine” described in [section 13.1 \(The Actual Redirect\)](#) will now be higher than what can be handled within the normal timeframe. If the Max. redirect speed is decreased to 50%, the time the browsers will receive the “FirstInLine” will be doubled.

The following can lead to very long waits in the “FirstInLine” status:

- Large decreases in Max outflow.
- A paused queue and a high number of re-entries into the queue followed by a running queue and a low Max outflow.

## 14 Queue-it Availability

Queue-it takes pride in our extremely high availability, high uptime, and fast response times. Using continuous integration, we hot deploy new versions and the service does not need to be taken down for maintenance. We also have data center and segment backups / fall-overs.

In the rare cases that we have issues or that the access to our service is limited from some global region, customers will always be able to see the current status on our platform on <http://status.queue-it.com/>

The CA monitor is connected to our health-check service (see the Queue-it Health Check White Paper) and only shows green if the monitors can connect to the health-check end-point when the service responds with OK on all our check-points.