RAPIDSPIKE®

SYNTHETICS
Critical journey monitoring

Synthetic User Journeys - **Technical Guide**

AN IN-DEPTH GUIDE ON HOW TO USE SYNTHETIC USER JOURNEYS TO EASILY FIND AND FIX PERFORMANCE ISSUES AND SAVE ON MONITORING COSTS.

WHAT IS A SYNTHETIC USER JOURNEY?

A Synthetic User Journey is a path that a real user takes when using your website or app. Most websites are built to allow users to complete one or more key goals. For some sites this might be to generate sales leads via a feedback form, for others, it might be to sell products via a catalogue, shopping cart, and payment gateway.

For each goal, we can define the series of steps a user would take to complete it and from this create a unique user journey monitor. This monitor can be run regularly at any hour, from anywhere in the world. The monitor behaves like a real user - using a browser that downloads page elements, pauses between actions, looks for content and interacts with buttons and form elements. User journeys can be configured to follow any sort of key application process, from simple browsing of a website to complex interactions, communicating with an API, and much more. For this reason, every User journey scripted by RapidSpike is bespoke and tailored to your requirements.

WEBSITE PERFORMANCE STATS:

40% of users abandon a website that takes more than 3 seconds to load.
AKAMAI

75% would not return to a site that took more than 4 seconds to load.
AKAMAI

47% of users expect a web page to load within 2 seconds or less.
AKAMAI

Every 100ms of page latency reduced Amazon's sales by **1%**
AMAZON

1 second delay in page response can reduce conversions by **7%**
TAG-MAN

Each second cut from load time increased overall conversions by **2%**
WALMART

SYNTHETIC USER JOURNEY BENEFITS:

User journey monitoring combines usability testing and performance analytics to ensure website users can complete goals, as well as produce statistics around these steps, actions, and goals. User journey performance statistics show how long different goals take on your site. User journeys can be tested from different locations around the world, 24/7. There are many benefits to using synthetic user journey monitoring:

Visualise your customer journey performance:
- Understand your customer journeys through video & deep technical data.
- Performance tuning for better business outcomes based on customer behaviour.
- Optimize every single page in your customer journey with actionable insights.
- Minimise disruption for you, your operations team, and your customers.

See issues before they become issues for your customers:
- Watch your critical user journeys unfold on dashboards in TV mode.
- Pinpoint issues and drill down to their root-causes to resolve them quickly.
- Proactively manage customer experience and prevent customer basket loss.
- Get alerts fast using: Pagerduty, Opsgenie, Slack, Teams, SMS & more.

WHY MONITOR A USER JOURNEY?

Monitoring user journeys allows you to take a deeper look into the performance of your website or web application. User journeys help you to create a baseline of performance so you can identify when your app starts to slow down, outlining emerging problems early so that outages can be avoided. Many on-page errors are hidden from the naked eye and are usually identified by accident or by a user. The user journey monitor continually identifies these errors providing the web team with detailed insight into the inner workings of the website or web application. Most websites and web apps rely on third-party hosted plugins or software, and the slowing down or failure of these elements can have a detrimental effect on user experience. User journey monitoring will identify performance issues and critical failures in these elements.

COMMON SYNTHETIC USER JOURNEYS:

Shopping Cart & Checkout
Browsing for products, choosing sizes and adding to basket and proceeding through the checkout.

Account Registration & Login
Creating a new account, logging in to an application, viewing data.

API Transactions
Authentication, read and interact with an API endpoint.

Download PDF
Ensure downloadable assets can be delivered to customers via a download link.

Product Search
Use a website search tool to search and click on a product.

Check Regulatory Functionality
Check your website or app's functionality complies with industry regulations.

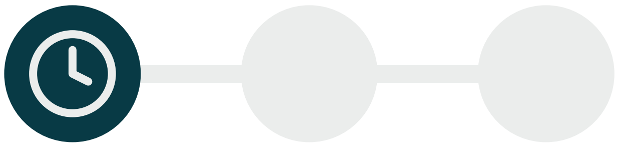
MONITORING STRATEGY DESIGN

There are a number of factors that should be considered when designing your monitoring strategy. Website hosting and user base locations, key business processes, internal requirements and your system architecture should all be discussed. Monitoring platforms often see people sign up and only use uptime monitoring, or create a single user journey that tests everything in one go. Whilst both valid uses, each has its downfalls and monitoring blind spots that the other covers. It is important to monitor a website in several ways to have a true representation of your website from the customers' perspective.

UPTIME MONITORING

Uptime monitoring is the fastest and most global type of monitoring RapidSpike offers - you can run checks as frequently as 1-minute from multiple locations on almost every continent. Therefore uptime should be the bread and butter of your monitoring strategy. Uptime monitoring (using HTTP GET) will detect and alert whether your pages are accessible, check the status code they have returned, response time from the page itself and check specific content (exists or not exists). Additionally, Ping and TCP monitors can give a response time and top-level availability of your hosting server or network device.

Run monitors quickly and from as many locations as are relevant to your website hosting location and your user base. You should also consider layering in content checks that will check for the existence (or non-existence) of specific HTML in the response we receive from your target URLs. RapidSpike leverages our global network to double-check failing uptime monitors from other locations around the world before we notify you – so you can be confident that we haven't notified you without good reason.



PERFORMANCE MONITORING

Performance monitoring gives you the ability to add alert rules to help determine the actual health of your webpages. A vital part of your complete monitoring package, by monitoring the parts of your pages that uptime monitors don't track. This includes major customer-affecting issues like poor performance and on-page errors.

For example, you could configure alerts based on the elements that load into your pages and therefore notify you when any stop loading or take longer than you'd like them to. Additionally, you could set an alert to trigger if any images grow to larger than a predefined size. This is particularly useful if people in your organisation are able to upload images to your website without compressing or resizing them. Alerts such as these can help prevent slow-loading pages before they are spotted by your end users.

Performance monitoring cannot be run particularly frequently and should only be run from a small number of regions, therefore they should be thought of more like interim health checks rather than used for uptime/downtime alerting.

SYNTHETIC USER JOURNEY MONITORING

User journey monitoring has the ability to check everything uptime and performance monitoring can, and more. Identify slow elements, faulty third-party scripts and problem pages on your app. Track how software and design updates affect overall speed and use this data to improve customer experience. Similarly to performance monitoring, user journeys run from limited locations and at slower frequencies than uptime and therefore shouldn't be used as your only uptime/downtime monitoring tool.

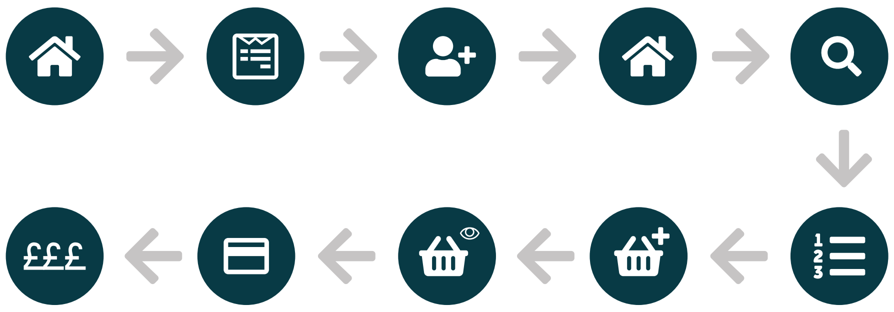
RapidSpike offers video user journeys that provide their own benefits to regular user journeys. This feature makes it easier to identify issues clearly, and therefore resolve them faster – keeping your website running smoothly. The video is a full recording of your journey, as a user would see it. Pause and play at any moment to inspect the journey in greater detail and compare with the test result's screenshots. Pop-ups throughout the video show each journey action as they're being run.

ATOMIC JOURNEYS

The temptation with user journeys, is to create a single, long running journey that checks everything in one go and run it every 5 minutes. This may sound like a good idea because one journey is cheaper than five or because there's fewer scripts to maintain, but it will make your life much harder and is likely to still cost as much due to the total time it takes to run. Therefore a good approach to user journey monitoring is to create atomic journeys.

The process for defining journey flows is not too dissimilar to normalisation theory in database design. You start with the main requirements in their fullest form, then iteratively break that down into smaller, atomic components. This is where the term 'atomic journeys' comes from. To illustrate this technique, let's say you start with this flow called "New User Product Search, Checkout and Payment" and you wish to run it every 5 minutes from 1 region:

- Visit homepage
- Click "Register"
- Register a new user
- Return home
- Search for a product
- Select a product from the results
- Add the product to the basket
- View the basket
- Go through the checkout process
- Submit dummy payment



This script will take a long time to write, touch many parts of the target website and will require in-depth knowledge and a lot of care to maintain. It also produces 288 results per day (12 per hour). So the first step here would be to break it down to its main requirements:

- New user registration
- Product search
- Add to basket
- Checkout process
- Payment check

Depending on the architecture and behaviour of the target website, these 5 requirements could each be their own journey, however, this is unlikely to be the case as the payment check would probably require a checkout process to get to it. Also, depending on how the inventory works, adding an item to a cart inside an authenticated user session might tie that item to an account for longer and therefore use up inventory that real people might want.

So, a new set of journeys would be proposed (including frequency and subsequent results per day):

- Once daily (1/day): New User Registration
  Start on the homepage and navigate to the registration page.
  A new user is registered on the website.
- Every 3 hours (8/day): Product Search
  A search term is entered on the homepage and submitted.
  The results are validated against the search term used.
- Every hour (24/day): Direct Add to Basket
  Start on a highly available product's page.
  Add the item to the basket and verify the success.
- Every 15 minutes (96/day): Existing User Login
  Start on the homepage and navigate to the login page.
  Enter and submit test login credentials then verify the success.
- Every 15 minutes (96/day): Add to Basket and Checkout Process
  Login to an account with a product already in its basket.
  Start the checkout process and work through it.
  Attempt a dummy payment.

This final strategy will mean the target is tested for its most important functionality at least every 15 minutes, with other functionality checked less frequently to preserve costs. The total results produced would be 225, therefore cheaper to run.

If your customer base spans two continents then the temptation with the first strategy would be to increase the test regions to two. This doubles your results produced and therefore dramatically increases costs. The proposed strategy would enable you to set some journeys to one region and some to another, thus keeping the results output and cost the same, but increasing the geographical test coverage.

BENEFITS OF AN ATOMIC JOURNEY STRATEGY

- Faster script writing - lower test times mean the build process is quick.
- Faster fault finding - atomic journeys make fault finding quicker.
- Lower maintenance overhead - fewer components means fewer things to juggle.
- More regional coverage - each journey can run different test regions for no extra cost.
- Better test coverage to cost ratio - this strategy produces 225 results per day from up to 5 regions, whereas a single 5 minute journey in 1 region would produce 288.

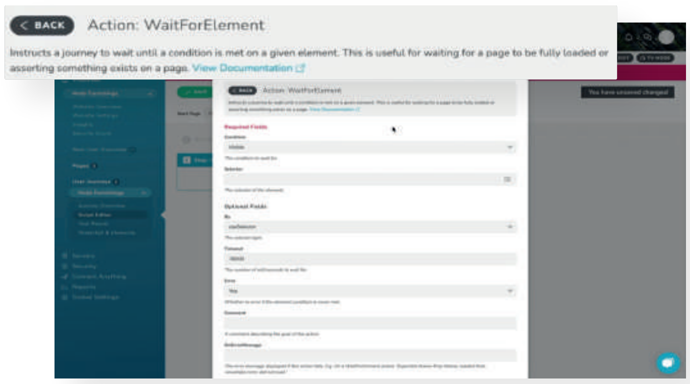
SHOPPING BASKET MONITOR

The ‘Add to Cart’ user journey is one of the most common user journeys used by ecommerce owners. For obvious reasons, ensuring a customer can search, select a product, and checkout successfully is a vital function. Synthetic user journey monitoring can assist in creating a positive user experience by tracking performance metrics, and by being alerted if issues do occur.

The steps of the average Homepage to Checkout Journey are:

Step 1: Load the homepage	Step 2: Search for a product	Step 3: Select a product	Step 4: Add product to cart	Step 5: View cart	Step 6: Checkout process	Step 7: Submit dummy payment

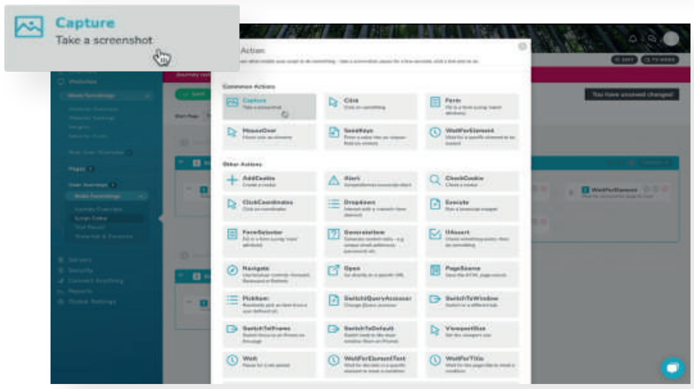
Step 1: Start your journey by using a Waitforelement action to acknowledge the page has loaded successfully. To find the selector for an element you would like a user journey to interact with, load the webpage and right click on your chosen element. Use the “Inspect” tool from the menu that appears. This loads the code of the website in a pop-up box, which you can through to find the selector you need. Follow this with a Capture action for visual confirmation.


Action: WaitForElement

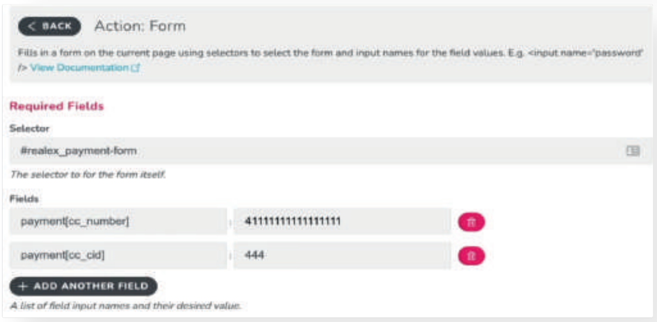
Instructs a journey to wait until a condition is met on a given element. This is useful for waiting for a page to be fully loaded or asserting something exists on a page. View Documentation

Step 2: Begin with a click to load the next page of the website, such as a product page. Use a Waitforelement action and a Capture to confirm this, and then you can interact with the elements on the page, such as selecting sizes and colours of products. For example, you can click an ‘add to basket’ button.

Step 3 + 4: Repeat the same process as step 2, where select a product and then add it to the basket. Beginning with a Click, Waitforelement and Capture to load the page, selecting add to basket. It’s advisable to manually replicate the journey whilst you build it in the RapidSpike app, to ensure you account for any pop ups or dynamic elements.



Step 5: Fill in an email input box and an address form. For the input box, you can use a Sendkeys action to fill in a single input box. To do so, find the selector for the email input and enter this into the Sendkeys action in the script builder. Whereas to fill in the address form, use the Form action. Find the selector which highlights the whole body of the form, as well as the name attributes for individual input box within that form. To interact with the address finder, use the Dropdown action. Find the selector for the dropdown menu, and then in the script builder, you can enter either the selector or value of the option you would like to choose.


Action: Form

Fills in a form on the current page using selectors to select the form and input names for the field values. E.g. <input name="password" /> View Documentation

Required Fields

Selector: #realex_payment-form

The selector to for the form itself.

Fields

payment[cc_number]	4111111111111111
payment[cc_cid]	444

A list of field input names and their desired value.

Step 6 + 7: To fill in the card payment details you can use the Form action again. Click to place the order and then use a Waitforelement action to wait for the error message in response to the dummy card details, and use a Capture to round the journey off. When you have finished writing your user journey in the script builder, use save and check to run a one-off test to make sure everything is running smoothly. Finally, when you are satisfied with the journey, click publish.

CHECKING THE PAYMENT GATEWAY WORKS

Many of our clients ask us to write journeys to ensure their payment gateways are operational. This is one of the most key processes that can be required of a company's website and a broken payment form can cost a business a huge amount of money. Not only this, but the payment form can be the target of client-side attacks such as those from the Magecart hacker group.

Whilst monitoring these payment gateways may appear like a simple request, it's often one of the more complex journeys we are asked to write. To get to the payment gateway of an ecommerce website, for example, you would usually first need to make your way through a whole basket and checkout process. This alone is a lengthy process and can be made trickier by any number of factors along the way such as product option selection (size and colour etc), avoiding out of stock items and selecting delivery options.

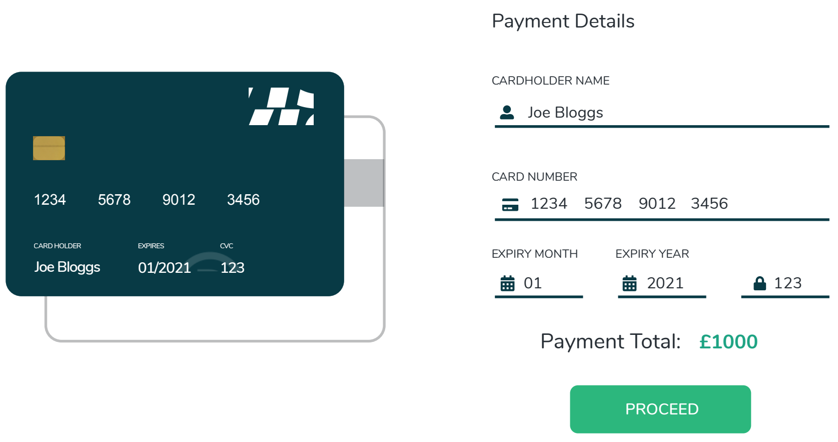
The problems still persist even when a payment gateway is successfully reached. To start with, we aren't able to store and test with real credit card details due to compliance regulations. So ideally the target website would be set up to accept dummy or fake details in order to simulate a real scenario and check the gateway properly. Also, if the website offers multiple payment gateway options (e.g. a native gateway, PayPal and Klarna) then multiple journeys may be required to achieve the desired test coverage.

Generally, our advice and best-practice solution to this requirement would be to take a leaf out of the 'atomic journeys' strategy discussed earlier. The idea would be to shorten the time taken to get to the payment gateway as much as possible. Techniques to use may include the following, but are largely dependant on the target website's behaviour or architecture:

- Selecting high-availability products that will not go out of stock.
- Randomising the product selection to reduce the chance of making a product go out of stock by reserving too many in test baskets.
- Add logic to be able to check for availability on product listing pages so only in-stock product selected.
- Use zero-option products such as accessories with no size options or furniture with no variants.
- Logging into special user accounts that already have active baskets setup with dummy products created solely for the purpose of testing and addresses already stored to reduce the number of interactions required.

WHY MONITOR THE PAYMENT GATEWAY?

Observing the way the payment gateway works before writing a journey is essential for ensuring the journey tests the payment gateway. If the website performs some client-side validation, that we know also checks the details in the gateway provider, then potentially only filling in the card details form may be enough to test it. However, if the gateway requires submitting then dummy, fake or pre-allocated card details will need to be used and the failure state checked by the journey. Monitoring the payment gateway is essential for any website with transaction functions. Ecommerce brands like Tryzens benefit from these tests to ensure their checkouts are up, performing as they should and are secure.



Payment Details

CARDHOLDER NAME	Joe Bloggs
CARD NUMBER	1234 5678 9012 3456
EXPIRY MONTH	01
EXPIRY YEAR	2021
	123

Payment Total: **£1000**

"With RapidSpike's User Journey monitoring we protect the brand of all our E-commerce clients by making their websites more reliable and secure. We also use the tool to help us make their websites faster for customers with less errors."

- Matthew Briggs, Senior Vice President – Live Services at Tryzens

USER JOURNEY BEST PRACTICES

To ensure your user journeys are performing efficiently there are some useful tricks to help. Firstly, try to use unique selectors, usually, element IDs are unique to the element on that page. This helps the user journey navigate through your website with greater precision. Other than the first step of a journey, which usually requires a Waitforelement action and a Capture action, it is best to start each new step of your user journey with a click to proceed onto the next page of the website. This means each step of the journey is on a different page of the website. Doing this means that the Waterfall, which provides a breakdown of the load times of the elements the journey encountered, will be divided by page too. This is useful to understand the load time of each page of your website.

RapidSpike user journeys move through websites very quickly, much faster than the average user. Use Wait actions and Waitforelement actions to slow the journey down so it behaves more like a real user. Use as many Capture actions as you can. Captures help you ensure your journey is running as expected. It is helpful to insert capture actions after key junctures in your journey, such as filling in a form, to ensure all the details have been entered correctly.

Top Tip: To get around recaptcha tests that block our user journeys, it is useful to safelist our list of IP addresses or to use custom headers. This prevents your site from recognising our user journey software as a bot. We can recommend using custom headers as this is easier to maintain than a long list of IPs.

USEFUL USER JOURNEY TOOLS:

IfAssert
To close pop-ups or cookie banners, or any features of a webpage that only appear on occasion, use our IfAssert action. This establishes whether or not an element is present on the page. If it is, you can insert actions to take the journey down one path and another if it is not.

iFrames
Use our Switch to iFrame action to interact with any elements inside iFrames. Remember to follow this with the Switch to Default action to return to the code on the webpage outside the iframe.

Mouse Over
This action allows you to hover over elements on a page.

Execute
Use this action to execute a snippet of JavaScript.

Sendkeys, Forms and FormSelector
The Sendkeys action is useful for single input boxes, such as a search bar. The Form action will fill in a form using the name selectors for the different input boxes within the body of a form. The FormSelector action uses the selector of each input box within the body of a form to fill in information.

Dropdown
Use the selector of the dropdown, and either the value or selector of your option choice to fill in dropdown menus.

WaitForElement
This enables your script to be more resilient to performance issues and helps with debugging problems before they manifest elsewhere.

Navigate
Navigate backwards, forwards, or refresh.

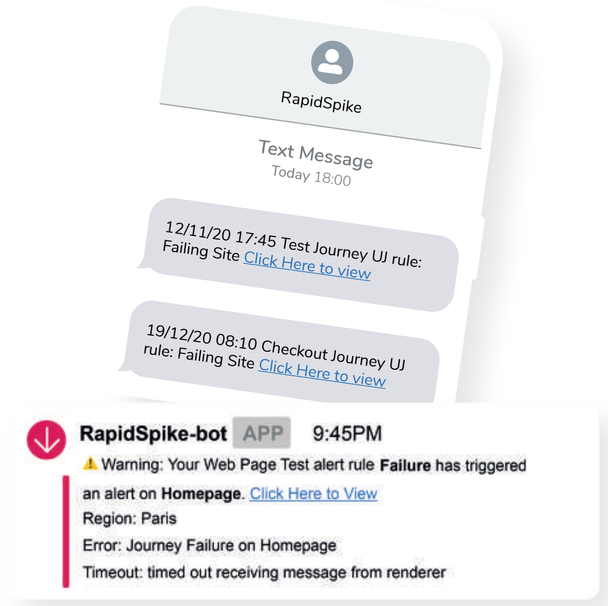
ALERTS

The final stage to defining your monitoring strategy is to understand what your baseline statistics are and therefore "what is normal". Normal will depend largely on the business requirements of your journey monitoring and what your motivations are. Alerting options range greatly from simply checking the functionality of the target website to alerting on quotas being reached for individual page load or component size.

Another aspect to consider at this stage are the recipients and escalation processes that you need to build into the alerting plan. Simply sending everything to one place is fine if you're not expecting to receive a lot, but noise can quickly introduce alert fatigue and therefore raise the time to find and fix problems.

If you do have specific page size quotas to meet or are interested in image sizes on particular pages then these are worth defining before you decide on the journey path itself. In RapidSpike, a journey step should relate specifically to a single page on the target website. Therefore the way a journey is constructed is instrumental in partitioning performance and element data to the page they belong to. This then filters to your alert rules because you can assign them to specific steps (and therefore specific pages).

Once the journey is written and stable (this may take some iterations to achieve) then we always recommend allowing the journey a minimum of 48 hours to gather a good set of data. This is where you can learn what normal looks like to you. A step load time of 5 seconds may be expected if the step is submitting complex search criteria to an API, but that same time to load a product page will be troublesome.


RapidSpike

Text Message
Today 18:00

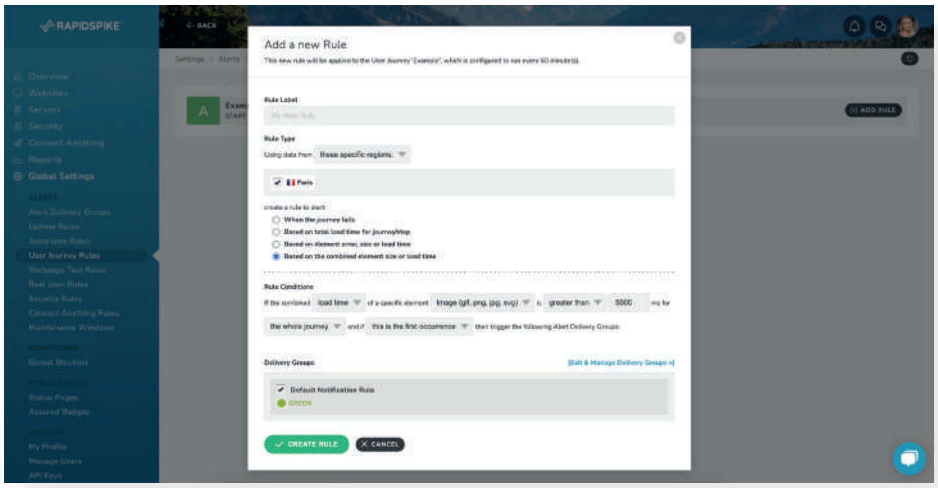
12/11/20 17:45 Test Journey UJ rule: Failing Site Click Here to view

19/12/20 08:10 Checkout Journey UJ rule: Failing Site Click Here to view

RapidSpike-bot APP 9:45PM
⚠ Warning: Your Web Page Test alert rule **Failure** has triggered an alert on **Homepage**. Click Here to View
Region: Paris
Error: Journey Failure on Homepage
Timeout: timed out receiving message from renderer

ALERT RULES

Once you have established your expectations for what normal is - whether this an element's load time, the size of a file, or if you just want to be alerted if your journey functionally fails - navigate to "User Journey Rules" under Global Settings in the left blue menu. Here you can establish rules for when your alerts should be triggered.


Add a new Rule

This new rule will be applied to the User Journey 'Example', which is configured to run every 60 minute(s).

Rule Label

Rule Type

Using data from: these specific regions:

✓ Paris

create a rule to alert:

- When the journey fails
- Based on total load time for journey/step
- Based on element error, size or load time
- ● Based on the combined element size or load time

Rule Conditions

If the combined load time of a specific element Image (gif, png, jpg, svg) is greater than 5000 ms for the whole journey and if this is the first occurrence then trigger the following Alert Delivery Groups:

Delivery Groups [Edit & Manage Delivery Groups »]

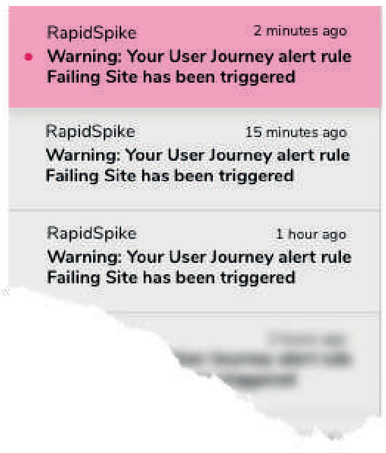
✓ Default Notification Rule
● GREEN

For synthetic user journey monitoring, you can set alerts for when the journey fails, based on total load time for a journey or step, based on an element error, size or load time; or based on the combined element size, or load time. You can customise the sensitivity for each of these alerts, for example, you can customise the exact load time for an element based on it's "normal" performance. You can also choose to be alerted if an incident occurs once, or multiple times, depending on the level of surveillance you would like to have over your journey. The granularity of RapidSpike's alerting system allows you to personalise the way your alerts are run and set your own priorities.

For each alert rule, you can link it to one or more alert delivery group - depending on who you would like to be alerted to the error, and the severity of the issue. Alert delivery groups are a way of saying who will be alerted and how. An extreme example of how to use these groups would be to route alerts to different support teams based on which region the journey triggered the alert rule in. For example, you may have a US support team who would want to know about problems encountered from a US test region, whereas the EU support team would not need to know about these.

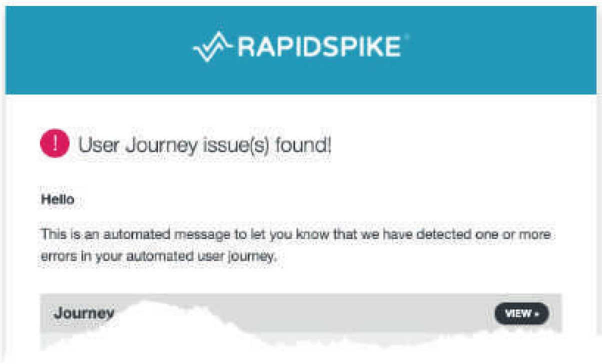
Your alerting strategy should also take into consideration your wider monitoring strategy, as discussed earlier. Uptime alerts should be set up for precisely that purpose - uptime alerting. Whereas journey alerts are designed for functional and guard rail alerting to protect the experience of your users.

RapidSpike offers a variety of alert delivery methods from SMS messages, Slack channel alerts, voice calls, and a number of integrations with tools such as Webhooks, Zendesk and Opsgenie.


RapidSpike 2 minutes ago
Warning: Your User Journey alert rule Failing Site has been triggered

RapidSpike 15 minutes ago
Warning: Your User Journey alert rule Failing Site has been triggered

RapidSpike 1 hour ago
Warning: Your User Journey alert rule Failing Site has been triggered


RAPIDSPIKE

User Journey issue(s) found!

Hello

This is an automated message to let you know that we have detected one or more errors in your automated user journey.

Journey VIEW »

RAPIDSPIKE SYNTHETIC USER JOURNEY MONITORING

RapidSpike can help monitor your user journeys by performing the same actions as real users, repeatedly throughout the day and starting from various geographical locations around the world. These *synthetic users* follow a pre-determined script designed to interact with key processes in your app - such as searching for products, buying services, and completing registration forms. This script allows for very complex and interactive user journeys to be configured rather than the same static test every time. The journey is then run from one or more locations anywhere in the world, and as often as required, helping you to build up a picture of your performance over time. As the journey operates you will receive alerts if page elements are missing, your journey takes too long, or if the script is unable to complete one of its steps.

WHAT DATA DO WE COLLECT?

Screenshots & Videos

Screenshot each page during the journey, or receive a video showing you the exact state of your app at each step.

HAR Files

HTTP Archive (HAR) files in a 'waterfall' view, showing speed, size, request and response headers of every element.

Geo Tagging

Geo-locate all elements based on IP Address and displayed in an interactive map view.

Performance Baselines

Track ongoing performance and benchmark your app speed, giving powerful performance data analysis.

Page Interactions

Check for specific content - such as a search result or heading - and automatically alert if it's missing.

Elements Lists

A sortable, filterable list of all elements loaded during a journey so you can identify by location, speed and size.

Threshold Errors

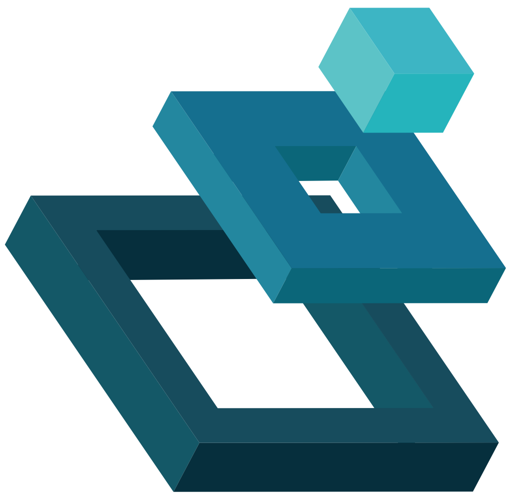
Set a speed threshold for the journey and get automatic alerts when this is breached.

Element Errors

Receive alerts when errors occur on any on-site page element or third party plugin loaded during the journey.

RAPIDSPIKE ATTACK DETECTION

In addition to ensuring your critical journeys are working correctly, it's important to make those journeys secure. Client-side attacks such as Magecart attacks are the number one threat to ecommerce sites. RapidSpike Magecart Attack Detection reduces the average detection time of these attacks from weeks to minutes. If you do have other security flaws, RapidSpike can act as your last line of defence against this particular issue. When configuring the Magecart Attack Detection monitor, you can either protect everything or choose the areas of your site which are the most vulnerable. Magecart Attack Detection is made up of three layers of protection: Client-Side Security Scanner, Synthetic Attack Detection, and Real User Attack Detection.



01 - CLIENT-SIDE SECURITY SCANNER

Actively scans the code on your website looking for commonly used patterns that have been used in historic client-side attacks.

02 - SYNTHETIC ATTACK DETECTION

Continuously walks through the critical areas of your website. It will see any data sent to untrusted hosts as it browses.

02 - REAL USER ATTACK DETECTION

Collects where real website visitor's data is being sent. If a malicious destination is added to your website, RapidSpike will alert you.

BENEFITS OF MAGECART ATTACK DETECTION:

- Reduces the average time to detection from weeks to minutes.
- Turns every user into a data guardian for your organisation.
- Ensures no malicious destinations get added to your website without your prior knowledge.
- Detects website skimming, formjacking, and supply chain attacks.
- Clear evidence for the ICO that you have taken steps to defend yourself.
- Insurance against your third-party security failures.
- Continuously monitor for attacks 24 hours a day 7 days a week.
- Last line of defence in case of issues, errors, misconfiguration which allow an attack.
- Receive alerts of any issues in the format of your choice (Email, SMS, Slack, etc.).