



Akamai Brand Guide for Third-Party Use

v_2

Have any questions about our brand or permissions? Email us at brand@akamai.com.

Welcome

Akamai is a registered trademark, and we take great care when evaluating our business partnerships, vendors, and consultants, both for sponsorship and philanthropic activity. If you've received permission to use the Akamai name, logo, or any of our other legally protected trademarks, please follow our brand guide appropriately.

Our brand guide outlines the rules for using the Akamai brand. Please be sure to follow the guide and only use the approved brand assets we provide. By doing so, you help us protect our brand.

Please note that this guide does not grant you permission to use any Akamai trademarks, logos, or brand features. If you've submitted the Brand Permissions Request Form, we will respond to your request for permission at our earliest opportunity. If you are an Akamai vendor or partner creating content and assets for or alongside us, please do not use this guide. Reach out to your Akamai contact for the proper guide.

Have any questions about our brand or permissions? Email us at brand@akamai.com.

About Us

Every day, billions of people connect with their favorite brands to shop online, play games, share ideas, manage money, and so much more. They may not know it, but Akamai is there, powering and protecting life online.

With the world's most distributed compute platform — from cloud to edge — we make it easy for businesses to develop and run applications, while we keep experiences closer to users and threats farther away. That's why innovative companies worldwide choose Akamai to build, deliver, and secure their digital experiences.

Our suite of leading security, compute, and delivery solutions are helping global companies make life better for billions of people, billions of times a day.

Our Brand Voice

Our four personality traits:

- Visionary
- Optimistic
- Reliable
- Smart

Voice is how we express our brand's personality.

Everything we put out should be:

Customer-centric: leading with *what* we make possible, *then* how

Inclusive: showing respect for all individuals, cultures, and communities

Simple: nurturing everyday conversation with humans (not bots!)

Honest: building trust through transparency and authenticity

Table of Contents

[Click on a title](#)

01. Akamai Logo

02. One-Color Logo Usage

03. Clear Space

04. Logo Misuse

05. Co-branded Logo Lockups

06. Our Tagline

07. Typography

08. Color Palette

09. Edge Graphic 2.0

10. Code Textures

11. Perspective Code Textures

12. Legal and Trademark

13. Contact Us

Akamai Logo

The Akamai logo represents our brand. The logo consists of two basic elements: the blue wave-like symbol and the word “Akamai” (known as the wordmark). This is a legally protected trademark of Akamai Technologies, Inc.

The proportion and arrangement of the Wave symbol and wordmark have been specifically determined and should be used exactly as provided. The wordmark should never appear without the wave symbol.

The blue and orange positive logo shown on this page is the preferred version, and should be used whenever possible. The wordmark and symbol should remain crisp and legible in all sizes. For digital screens, the logo should be readable at all screen resolutions that apply to the intended application.

[Download approved logo files](#)



Minimum Size

When the standard logo is being used, it must be no less than 1" wide.



One-Color Logo Usage

The Akamai logo has alternate usages, including one-color reverse. These iterations should only be used when the primary two-color versions are impractical for a given usage or cannot be reproduced by the production method. No other variations should be created. On dark backgrounds, the logo is white.

01. Single-color logo in white on Akamai Blue
02. Single-color logo in black on Akamai Orange
03. Single-color logo in white on black

01



02



03



[Download approved logo files](#)

Clear Space

To ensure the legibility and impact of the Akamai logo, it should always be isolated from competing visual elements, such as text and supporting graphics, with clear space.

Clear space is considered as the absolute minimum safe distance around the logo. In most cases, the clear space should surpass this minimum specification to avoid overcrowding and aid in brand recognition.

The minimum clear space is equal to half the height of the logo, or half the height of the “wave” mark.

Minimum Clear Space
“X” equals half the height
of the Akamai logo



[Download approved logo files](#)

Logo Misuse

The appearance of the Akamai logo must be consistent. The logo should not be misinterpreted, modified, added to, or altered in any way.



1. Do not alter the scale of individual parts



2. Do not rotate



3. Do not add any effects



4. Do not stretch



5. Do not reverse colors



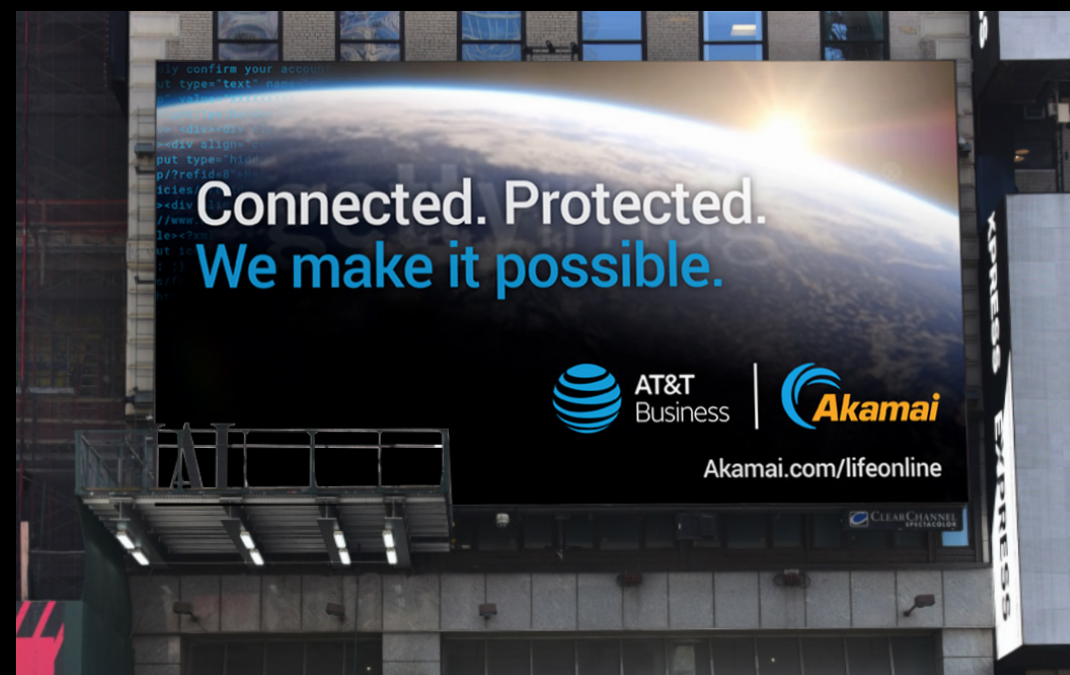
6. Do not remove the logo wave

Questions?

Email us at brand@akamai.com

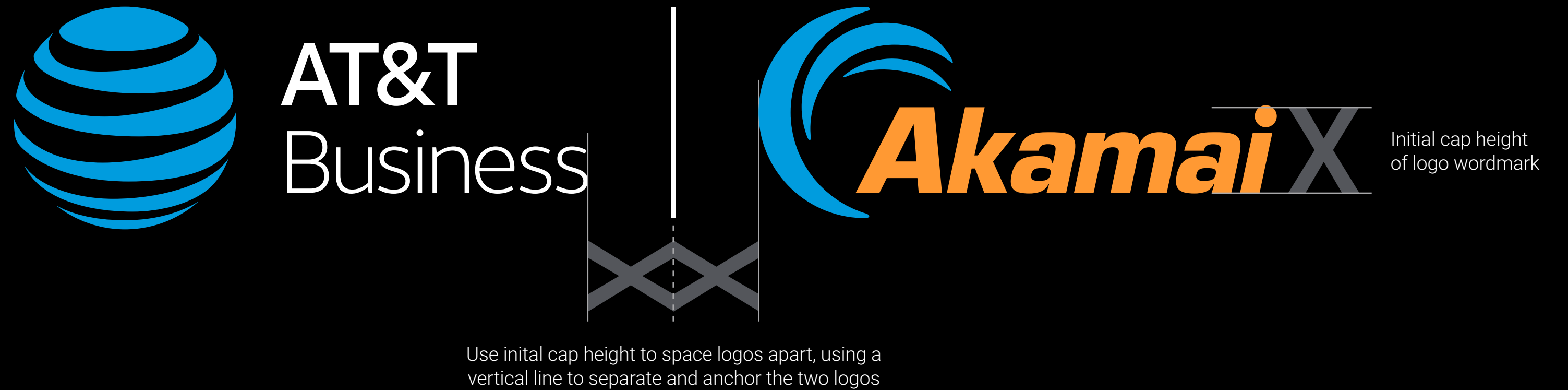
Co-branded Logo Lockups

We frequently partner with other companies to co-brand marketing initiatives, campaigns, or external events. When creating co-branded lockups for promotional purposes, try to maintain a balance between the two brand marks so that they have a similar visual weight. A simple vertical line should be used to anchor the two logos and provide a subtle but clear distinction between the logos.



Questions?

Email us at brand@akamai.com



Our Tagline

The Akamai tagline represents what we do and the massive impact we have on the world. It's also our mission statement.

You might have noticed that the tagline doesn't appear locked up with our logo. That's because we should be intentional about how we use it — as a prominent stand-alone statement, such as a headline on a web page, in a deck, or on event signage.



What we do:

- Cybersecurity
- Content delivery
- Compute

What we make possible:

Our impact on the internet and everyday life

Power and protect life online

Questions?

Email us at brand@akamai.com

Typography

Roboto Font Family

The primary typeface for the Akamai brand is Roboto. Chosen for its sleek, modern look, Roboto combines mechanical and geometric letterforms with friendly and open curves. This makes for a more natural reading rhythm more commonly found in humanist typefaces.

Roboto should be used within visual and written assets whenever possible as part of our overall brand expression across all marketing communications.

The Roboto font family is available in multiple weights and styles.

Questions?

Email us at brand@akamai.com

Character Set

ABCDEFGHIJKLM
 NOPQRSTUVWXYZ
 abcdefghijklm
 nopqrstuvwxyz
 0123456789
 @#\$%^&* / () { } []

Character Styles

Thin
Thin Italic
 Light
Light Italic
 Regular
Regular Italic
 Medium
Medium Italic
 Bold
Bold Italic
 Black
Black Italic

Color Palette

Primary Colors

Akamai Blue and Akamai Orange represent the two primary colors composing the logo. These colors are used in concert with the overall brand color palette to ensure brand recognition at every touchpoint.

Central to the preferred brand expression is the use of black as a foundation color. The consistent use of black, with Akamai Blue and Akamai Orange used as supporting colors, communicates a sense of strength and a premium look and feel to the Akamai brand.

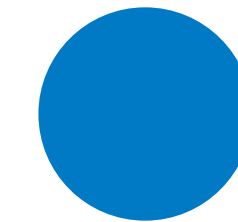
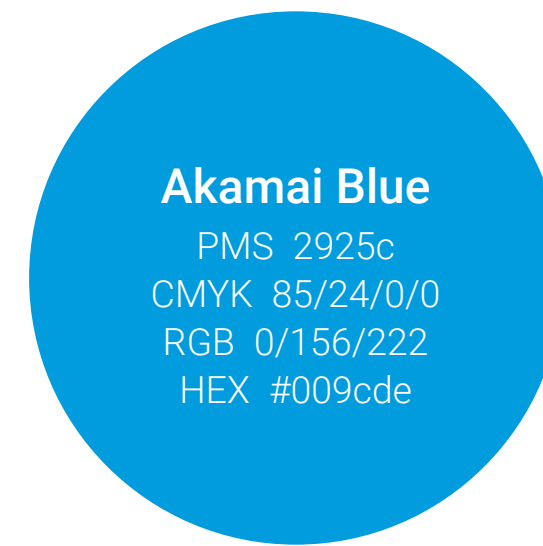
Secondary Colors

Each of the secondary colors may be used as a headline, an accent, or in certain cases, a background color. In general, the secondary colors should be used minimally to build a consistent hierarchy while providing flexibility and differentiation.

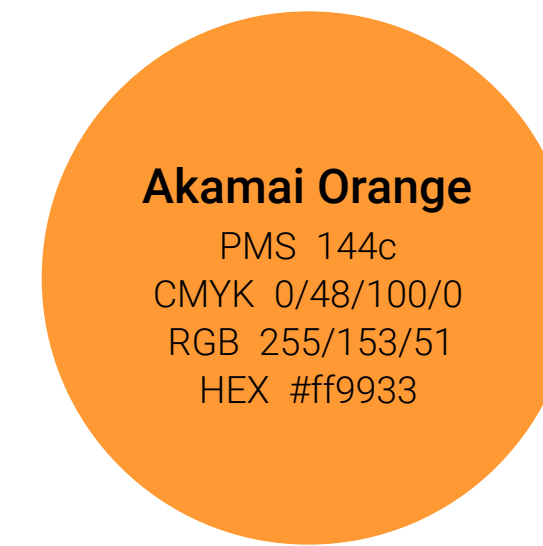
Questions?

Email us at brand@akamai.com

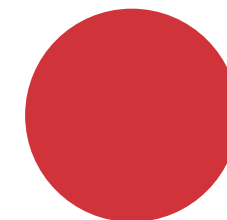
Primary Colors



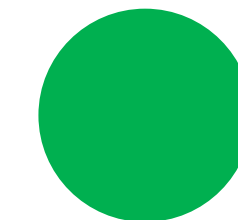
Websafe Blue
HEX #017ac6



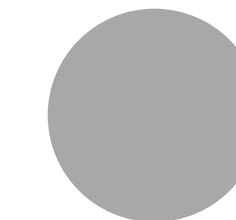
Secondary Colors



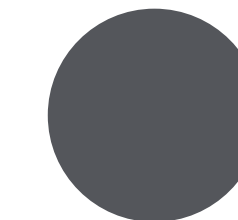
Red
PMS 1797c
CMYK 13/94/83/3
RGB 208/52/58
HEX #d0343a



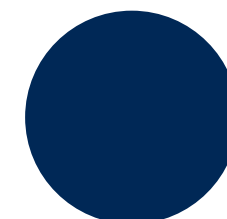
Green
PMS 354c
CMYK 95/0/90/0
RGB 0/176/80
HEX #00b050



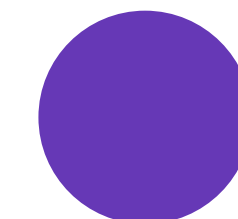
Light Gray
PMS Cool Gray 6c
CMYK 0/0/0/40
RGB 168/168/170
HEX #a8a8aa



Dark Gray
PMS Cool Gray 11c
CMYK 66/57/51/29
RGB 84/86/91
HEX #54565b



Navy Blue
PMS 295c
CMYK 100/84/36/39
RGB 0/40/86
HEX #002856



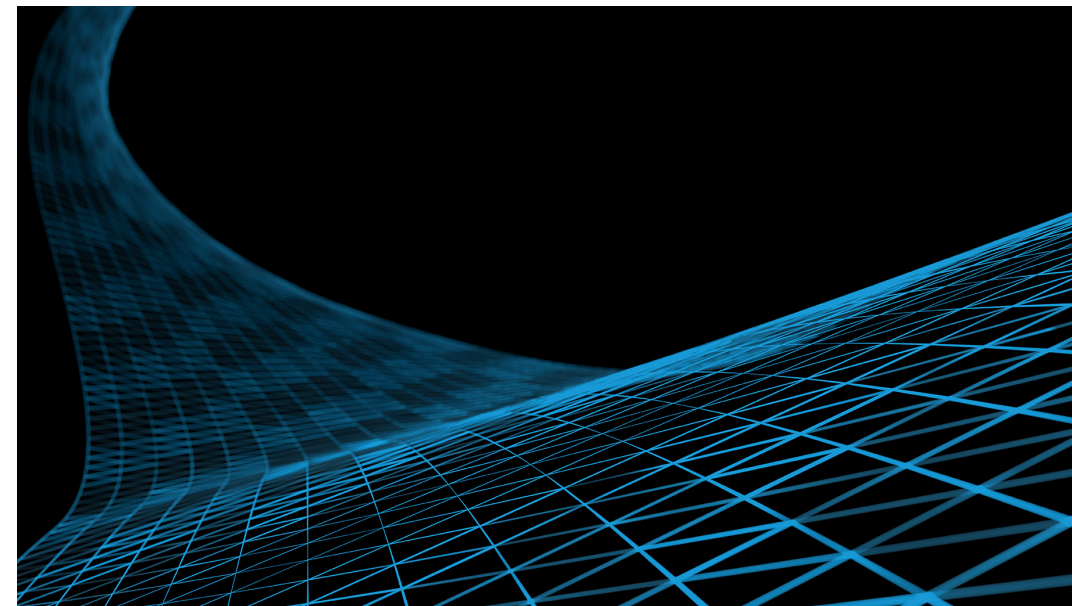
Purple
PMS 266c
CMYK 67/83/0/0
RGB 117/59/189
HEX #753bbd

Edge Graphic 2.0

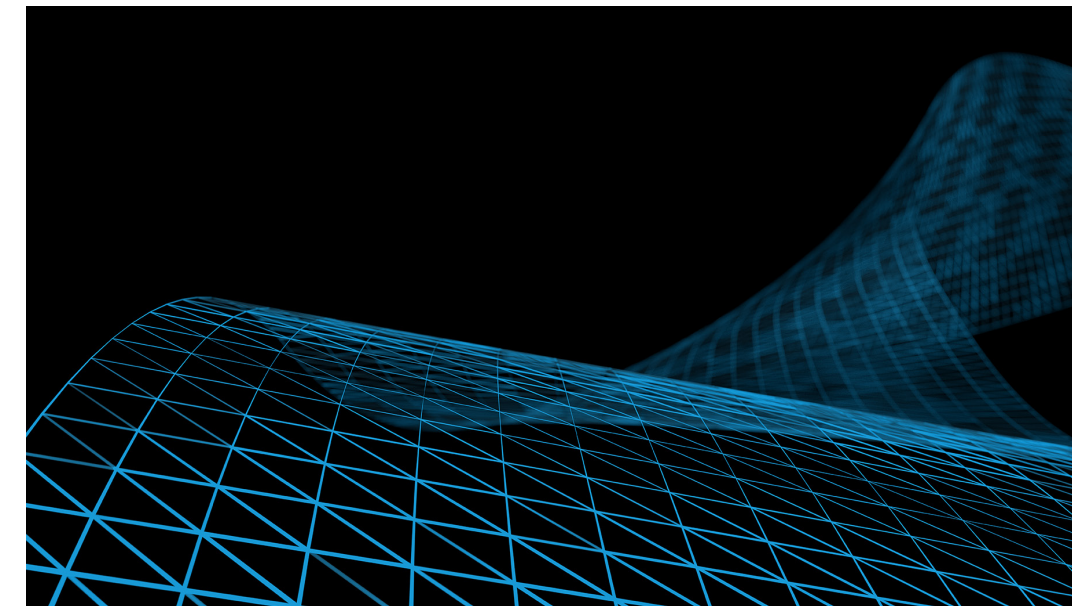
The graphic represents the Akamai Intelligent Edge Platform that surrounds and protects our customers' existing architecture.

Using the graphic in various ways, you can convey ideas like modernization, movement, and protection through the edge. Notice in these examples that a little goes a long way.

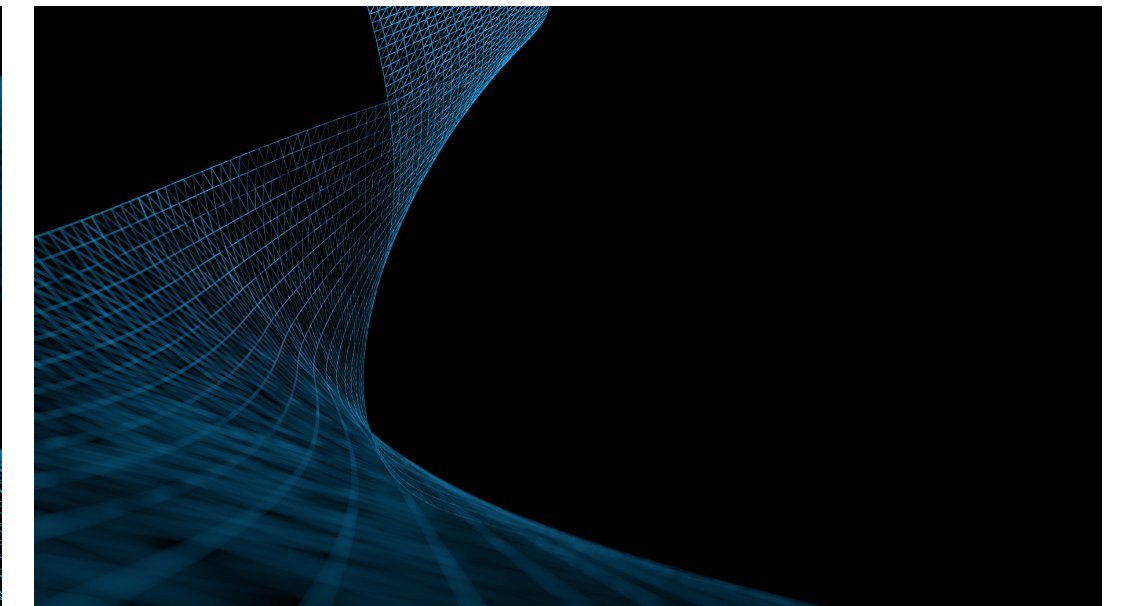
01



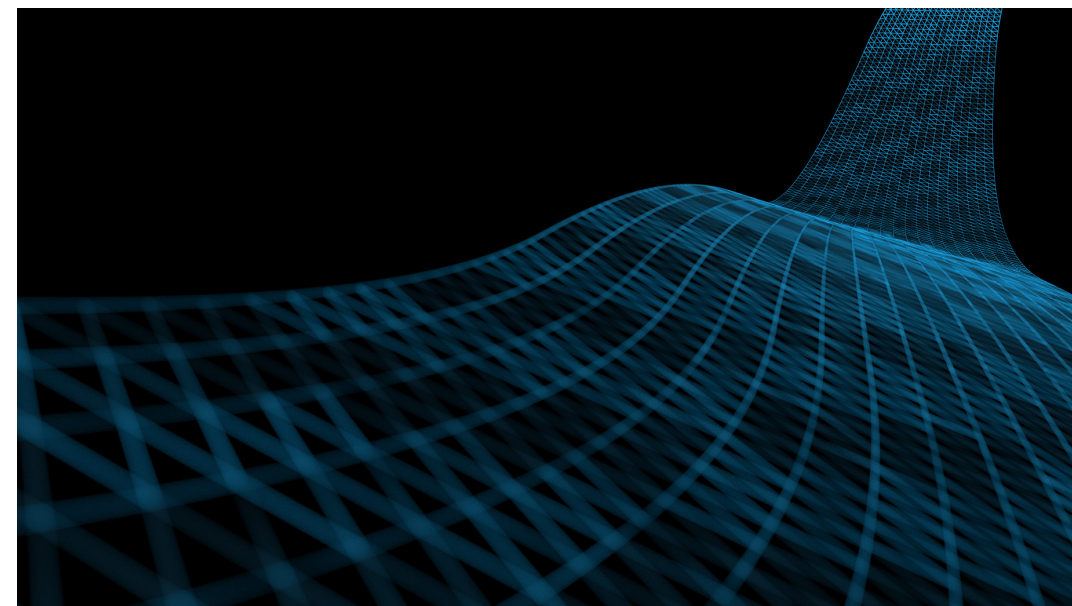
02



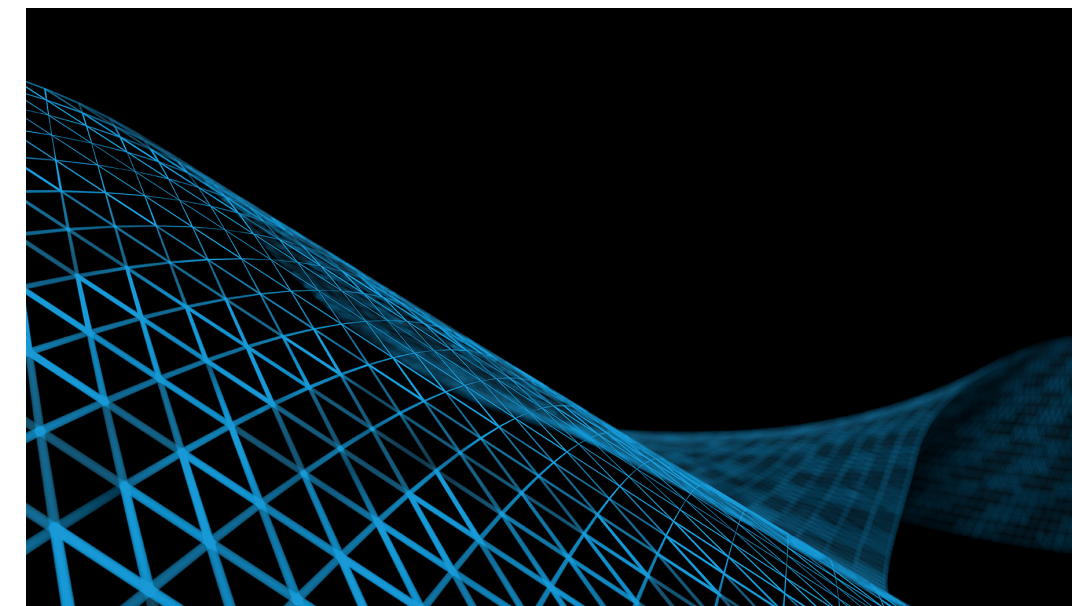
03



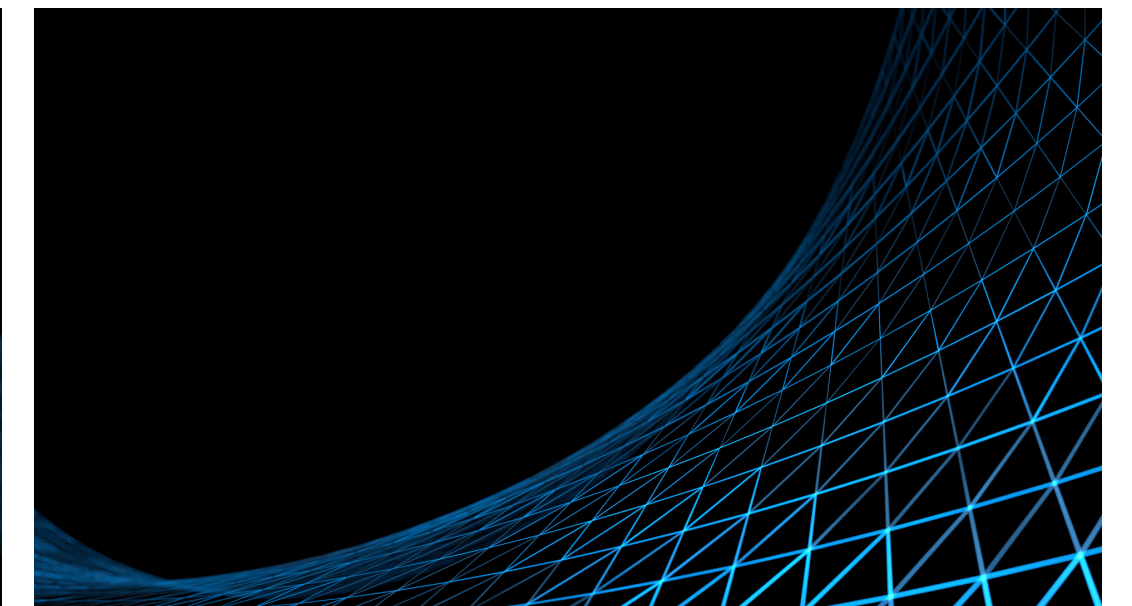
04



05



06



[Download approved edge graphics](#)

Code Textures

The code texture is an element we use as a thumbprint — a way of leaving an Akamai mark. Code represents the “how” of what Akamai makes possible, the unseen key to our algorithms, our uniquely intelligent platform, our pervasiveness in every part of the web, and the language that unites us with our customers.

Roboto Mono Medium is currently specified only for code treatments. Like the other members of the Roboto family, the fonts are optimized for readability on screens across a wide variety of devices and reading environments.

- 01. Akamai Blue code on black background
- 02. Akamai Blue code on blue background
- 03. White code on Akamai Orange background
- 04. Blue gray code on navy background

[Download approved code textures](#)

```

package main; import ( "fmt"; "http"; "strconv"
ControlMessage s
int64; }; func m
ControlMessage);
bool); statusPol
workerActive :=f
statusPollChanne

```

DDoS Code — Roboto Mono Medium

01

```

package main; import ( "fmt"; "html"; "log"; "net/
"strings"; "time" ); type ControlMessage struct {
}; func main() { controlChannel := make(chan Control
teChan := make(chan bool); statusPollChannel := ma
Active := false; go admin(controlChannel, statusPol
case respChan := <- statusPollChannel: respChan <-
<-controlChannel: workerActive = true; go doStuff
case status := <- workerCompleteChan: workerActive
(cc chan ControlMessage, statusPollChannel chan ch
Func("/admin", func(w http.ResponseWriter, r *http
strings.Split(r.Host, "."); r.ParseForm(); count,
FormValue("count"), 10, 64); if err != nil { fmt.
turn; }); msg := ControlMessage{Target: r.FormVal
html.EscapeString(r.FormValue("target")), Count:
tus", func(w http.ResponseWriter, r *http.Requ
bool); statusPollChannel <- reqChan; time.Sleep(
lect { case result := <- reqChan: if result == "
{ fmt.Fprint(w, "INACTIVE"); }; } else {
OUT");}); log.Fatal(http.ListenAndServe(":"
port ( "fmt"; "html"; "log"; "net/
ControlMessage struct { Target string; Count int64; };
Channel := make(chan ControlMessage); workerActive
statusPollChannel := make(chan chan bool);

```



03

```

package main; import ( "fmt"; "html"; "log"; "net/
"strconv"; "strings"; "time" ); type ControlMessag
Target string; Count int64; }; func main() { contr
:= make(chan ControlMessage); workerCompleteChan :=
bool); statusPollChannel := make(chan chan bool);
tive := false; go admin(controlChannel, statusPoll
for { select { case respChan := <- statusPollChan
Chan <- workerActive; case msg := <-controlChan
tive = true; go doStuff(msg, workerCompleteChan);
:= <- workerCompleteChan: workerActive = status
admin(cc chan ControlMessage, statusPollChannel c
bool) {http.HandleFunc("/admin", func(w http.Requ
r *http.Request) { hostTokens := strings.Split
r.ParseForm(); count, err := strconv.ParseInt
ue("count"), 10, 64); if err != nil { fmt.F
Error(); return; }; msg := ControlMessage{
ue("target"), Count: count}; cc := make
trol message issued for Target %s", Count: count,
tring(r.FormValue("target")), Count: count,
status", func(w http.ResponseWriter, r *http.Requ
reqChan := make(chan bool); time.Sleep(100 *
eout := time.After(time.Second * 10);
reqChan: if result == "INACTIVE"; }
Fprint(w, "INACTIVE"); }

```




02

```

package main; import ( "fmt"; "html"; "log"; "net/
"strconv"; "strings"; "time" ); type ControlMessag
Target string; Count int64; }; func main() { contr
:= make(chan ControlMessage); workerCompleteChan :=
bool); statusPollChannel := make(chan chan bool);
tive := false; go admin(controlChannel, statusPoll
for { select { case respChan := <- statusPollChan
Chan <- workerActive; case msg := <-controlChan
tive = true; go doStuff(msg, workerCompleteChan);
:= <- workerCompleteChan: workerActive = status
admin(cc chan ControlMessage, statusPollChannel c
bool) {http.HandleFunc("/admin", func(w http.Requ
r *http.Request) { hostTokens := strings.Split
r.ParseForm(); count, err := strconv.ParseInt
ue("count"), 10, 64); if err != nil { fmt.F
Error(); return; }; msg := ControlMessage{
ue("target"), Count: count}; cc := make
trol message issued for Target %s", Count: count,
tring(r.FormValue("target")), Count: count,
status", func(w http.ResponseWriter, r *http.Requ
reqChan := make(chan bool); time.Sleep(100 *
eout := time.After(time.Second * 10);
reqChan: if result == "INACTIVE"; }
Fprint(w, "INACTIVE"); }

```



04

```

package main; import ( "fmt"; "html"; "log"; "net/
"strconv"; "strings"; "time" ); type ControlMessag
Target string; Count int64; }; func main() { contr
:= make(chan ControlMessage); workerCompleteChan :=
bool); statusPollChannel := make(chan chan bool);
tive := false; go admin(controlChannel, statusPoll
for { select { case respChan := <- statusPollChan
Chan <- workerActive; case msg := <-controlChan
tive = true; go doStuff(msg, workerCompleteChan);
:= <- workerCompleteChan: workerActive = status
admin(cc chan ControlMessage, statusPollChannel c
bool) {http.HandleFunc("/admin", func(w http.Requ
r *http.Request) { hostTokens := strings.Split
r.ParseForm(); count, err := strconv.ParseInt
ue("count"), 10, 64); if err != nil { fmt.F
Error(); return; }; msg := ControlMessage{
ue("target"), Count: count}; cc := make
trol message issued for Target %s", Count: count,
tring(r.FormValue("target")), Count: count,
status", func(w http.ResponseWriter, r *http.Requ
reqChan := make(chan bool); time.Sleep(100 *
eout := time.After(time.Second * 10);
reqChan: if result == "INACTIVE"; }
Fprint(w, "INACTIVE"); }

```



Perspective Code Textures

The perspective code texture is a more dynamic take on our signature code treatment. The code should be readable, but should not compete with the headline and messaging.

The perspective code can be used at various angles, but should always fade to black. The amount of fade is determined by the copy and design elements overlaying the code texture. Make sure the copy and branding is always legible.

Primary Code Texture

01. Blue perspective code on black

Secondary Code Textures

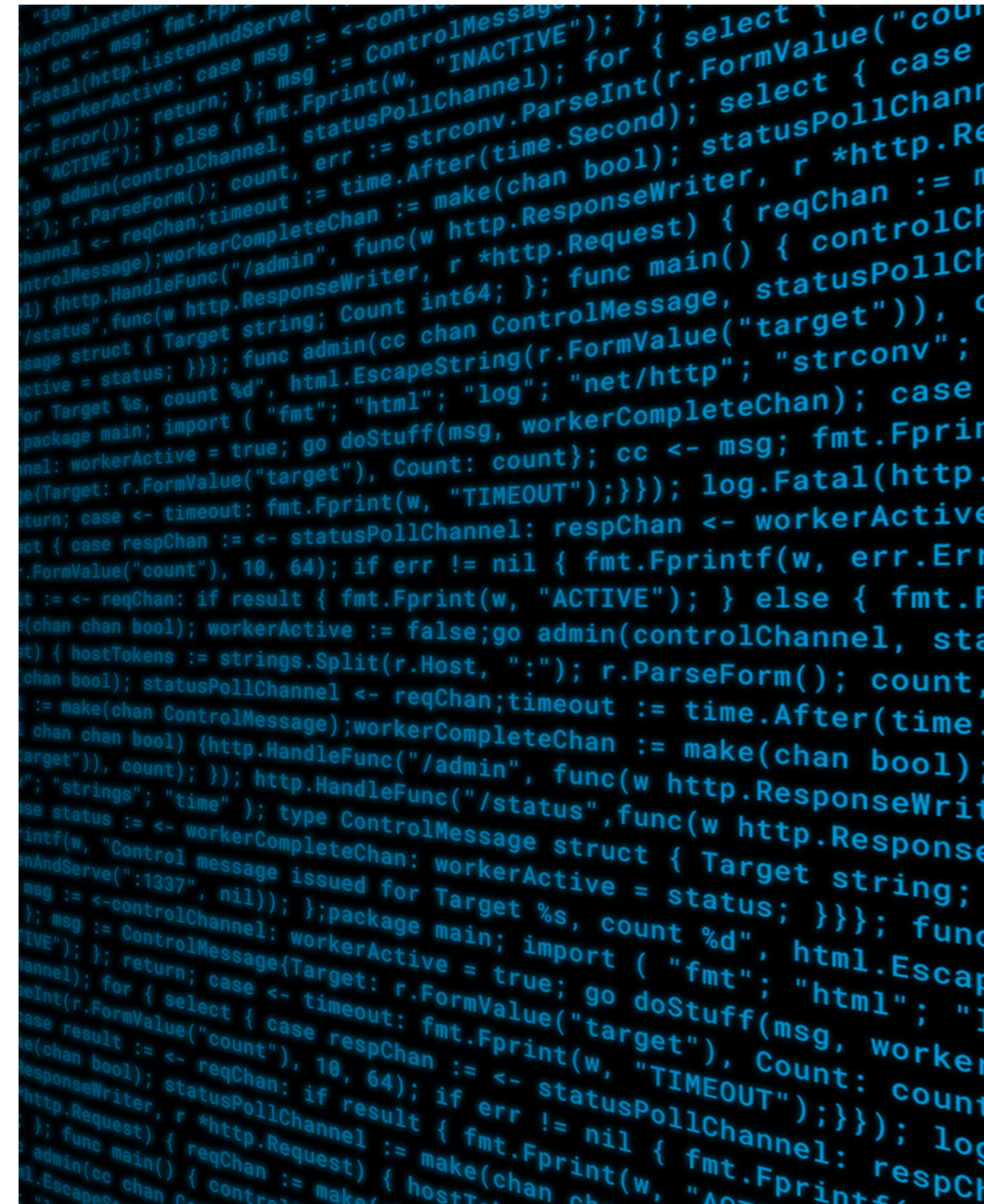
02. Orange perspective code on black

03. Green perspective code on black

[Download approved perspective code textures](#)

Primary Code Texture

01

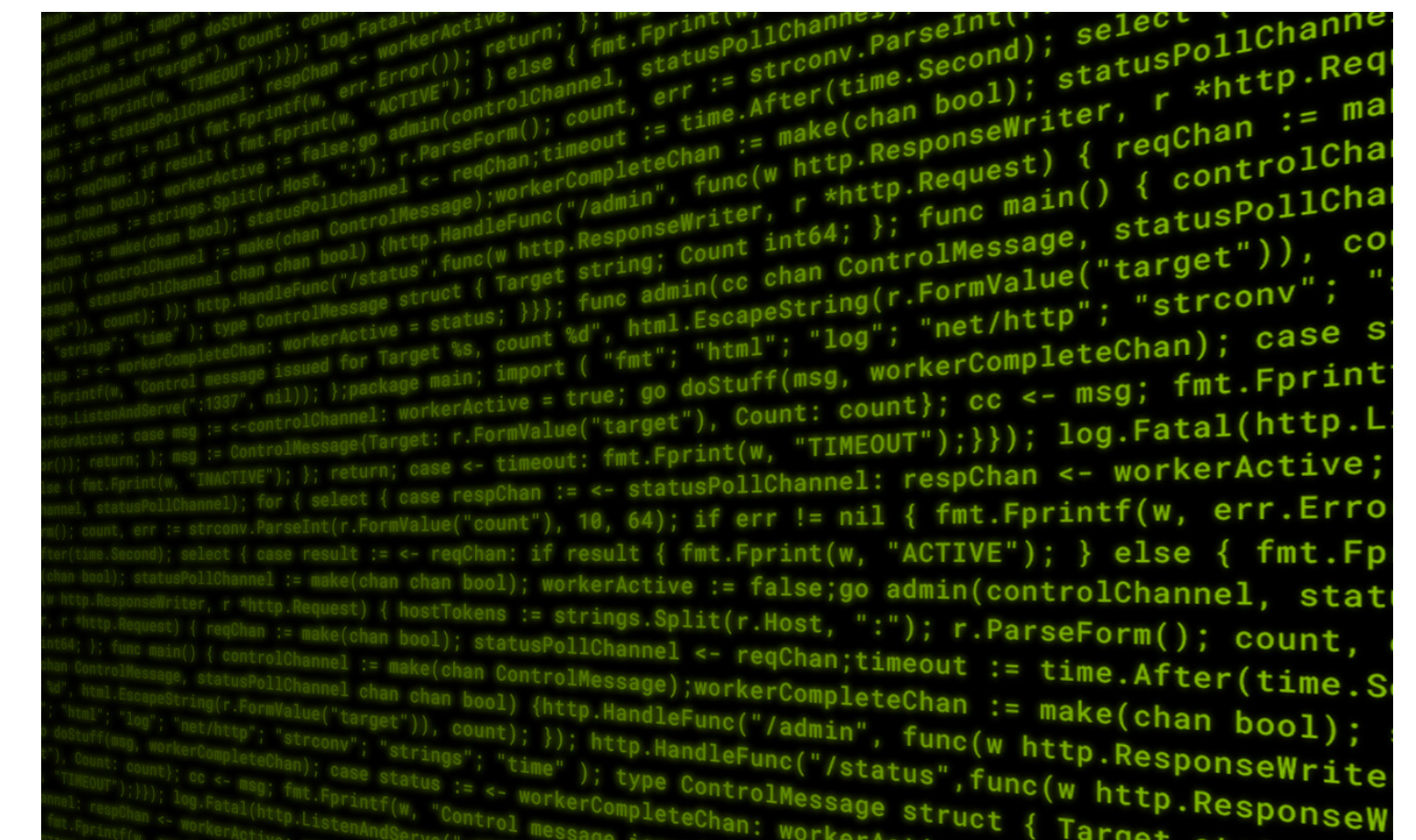


Secondary Code Textures

02



03



Legal and Trademark

To make things simple, we are generally no longer requiring the use of the ™ and ® symbols in the body of marketing documents or digital content when referencing Akamai terms that have registered or claimed trademarks.

Akamai and the Akamai wave logo are registered trademarks or service marks in the United States (Reg. U.S. Pat. & Tm. Off). Akamai Intelligent Edge Platform is a trademark in the United States. All use of the Akamai marks and all goodwill arising out of such use shall inure to Akamai's sole benefit.

Contact Us

Have any questions about our brand or permissions?

Please email us at brand@akamai.com.

Do you require layered assets?

Please email us at creative@akamai.com.

Found a website or any resource material that uses Akamai's trademark inappropriately?

We'd love to know! Please email us with your name and the material using our trademark inappropriately. We appreciate you taking the time to report this issue.