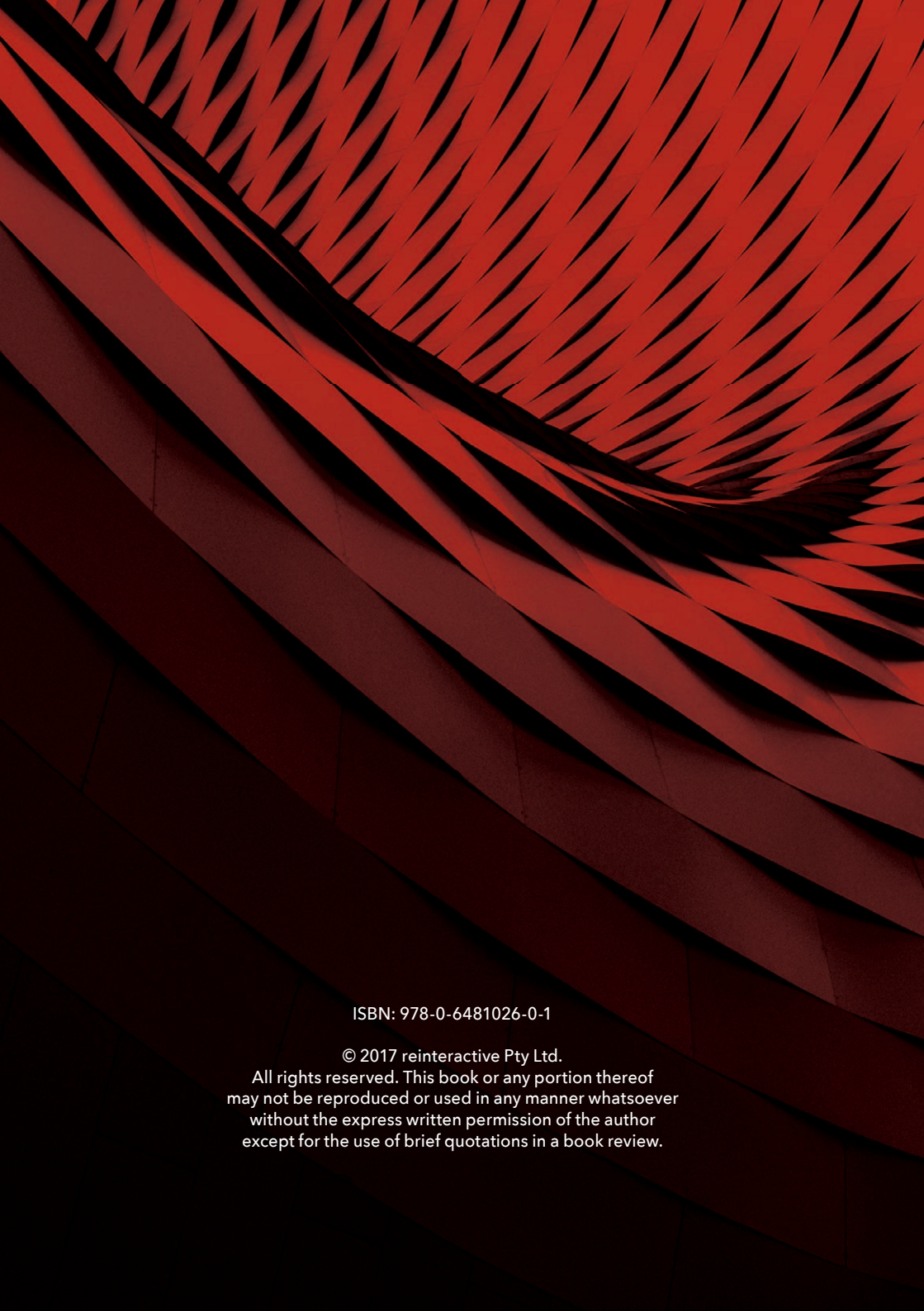


WHY RUBY ON RAILS?

HOW THE RIGHT SOFTWARE CHOICE
CAN SAVE YOU A TON OF MONEY

MIKEL LINDSAAR



ISBN: 978-0-6481026-0-1

© 2017 reinteractive Pty Ltd.

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the author except for the use of brief quotations in a book review.

WHY RUBY ON RAILS?

HOW THE RIGHT SOFTWARE CHOICE
CAN SAVE YOU A TON OF MONEY

by **Mikel Lindsaar**

a **re**interactive publication



Mikel Lindsaar • Founder of reinteractive

- Founder & CEO of reinteractive
- Founder & CEO of Exec.io
- Co-founder of Flood IO
- Author and Maintainer of the Ruby Mail Gem
- Ruby on Rails Commit team member
- Public speaker, author, serial entrepreneur
- Has served on the committees of Ruby Australia and the Australian Web Industry Association

About The Author

reinteractive's founder Mikel Lindsaar is well known in the Ruby on Rails community as the author of “mail,” the Ruby email handling library which has been downloaded over 90 million times and used by almost every Ruby on Rails installation in the world. He is known for his blogs and for his extensive work on the Ruby on Rails “ActionMailer” component. Mikel is a member of the Ruby on Rails commit team, an international team of peer-approved individuals who are authorised to make or approve changes to the core Ruby on Rails code base.

Mikel is passionate about this industry and fostering the growth, expansion and participation of the Ruby community in Australia and internationally. He has helped spearhead the adoption of the Ruby on Rails platform through his free Rails InstallFest initiative, introducing over 1,200 new developers to Ruby on Rails since 2014 almost a third of which have been female, helping promote diversity in a previously male-dominant sector.

Mikel has served on the committees of the Australian Web Industry Association Committee and Ruby Australia.

What is Ruby on Rails?

Before I go into why Ruby on rails is still the best choice for the vast majority of business applications, I'll quickly take a moment to explain exactly what Ruby on Rails is.

Ruby on Rails is an application development framework. Actually it's better known as a rapid application development framework, because it helps developers speed up the development of modern web based applications.

But what does “application development framework” actually mean? Well, a good analogy would be building a house. Every house is different and each one caters to the individual needs of the person that had it built. It would be pretty safe to say that no matter which house you walked into it would look different from every other house. But, walking into every house would still have similarities, things that were the same house to house regardless of their specific implementation.

For example every house would have several doors, these doors might look different, they might be painted differently or even made of different materials, but they all basically act the same as a door. Every house would have walls and windows and perhaps carpets and chairs and tables, all of these things would be common between houses even though they looked quite different in each house.

An application development framework provides similar

function. Every modern web application needs to access a database to store information, it needs to handle users logging in or resetting their password (when they forget) and it needs to be able to handle complex web pages with dynamic updates and so on.

Even though the way each of these is implemented might be different from application to application there is a common set of functionality to all of it. Ruby on Rails provides this common functionality so that a developer starting their developing can focus on what is important for the application – focusing on what makes it different from other applications, and save a lot of time not having to reinvent the common things that every application does.

By providing a simple way to do all the common things, Ruby on Rails was a pioneer in providing this common framework of functionality to web application developers, and so became known as an application development framework.

Because modern software applications are so complex, with many moving parts, pretty much every one of them is built on top of some Software Application Development framework. Which framework you use depends on what you are developing.

But I have found that if you are developing software for the web, or transaction or information based mobile applications, Ruby on Rails provides the most bang for buck assistance in getting you to market rapidly.

“There is nothing else out there that can develop the majority of business web based software as cost effectively as Ruby on Rails.”

Why Ruby on Rails?

A few days ago a prospective client asked “Why do you use Ruby on Rails?” and I told them a simple answer. Profitability and Productivity.

There is nothing else out there that can develop the majority of business web based software as cost effectively as Ruby on Rails.

This might be an unpopular position in an Information Technology (IT) world that releases a new and shiny tool every other day, but I don't care.

What I care about is the return for our customers, we could call this profitability. And the well-being of the team doing the project, which directly correlates to productivity.

Ruby on Rails provides both productivity and profitability, in buckets. Large buckets. Buckets you need a crane to shift.

This statement will no doubt raise comments a million ways to Sunday about how this is obviously false, with endless examples where a certain technology clearly trumps Ruby on Rails. And many of these edge case examples will be true, and I will agree with them completely.

But the assertion still stands; nothing else out there can develop the majority of business web based software as cost effectively as Ruby on Rails.

Now, I realise that I am talking from an obviously biased

position. Full disclosure time; I founded reinteractive. We develop business software that builds applications using Ruby on Rails. So I have a very clear, vested interest in saying that Rails is the best choice. So you can take this booklet with a grain (or a whole bag) of salt if you wish, but I intend to show you why I believe it to be true nonetheless.

First, am I qualified to make this assertion?

Perhaps not.

But, over the past decade my company and I have created, developed, worked on, debugged, hosted and maintained approximately 200 software projects all running primarily Ruby on Rails.

Across these applications we have bumped into a bunch of other technologies, including Python, Angular (versions 1 and 2), React, Ionic, various Apple tools, JQuery, Go, Phoenix, Elixir and plain old Javascript. We've worked on most of the major databases (both SQL based and non SQL based), as well as probably another dozen or so unmentioned and we've helped our clients deploy on many different hosting platforms. Additionally, I am in a position where I have been involved with almost all of these projects to some capacity from sales, oversight, customer management, and yes I'm happy to say, still coding!

I don't believe there are a huge number of people that have touched, looked at or coded on as many as

200 applications as well as working with all of those customers ranging from one person startups to some of the largest Enterprise and Fortune 500 companies.

One of the big takeaways from this experience is that in the vast majority of cases, the client just does not

“

What you want is something that a new developer can start on at 9am and could contribute meaningful code before she goes off to have her lunch.”

care if the app is using the latest and greatest technology or if their app responds in 100ms or 5ms. What they care about is getting a working system in production generating the intended revenue or savings as rapidly as

possible while maintaining a good level of quality and reliability.

Ruby on Rails delivers this.

There is a lot of software being written in the world and most of this is some sort of business automation or workflow management software. It is the software that pays the bills, its original and continued development could be said to be the foundation of the entire IT industry. The sole reason the IT industry exists is to make business and organisations operate in a more efficient, and therefore, profitable manner. Everything in IT does this. From the server rooms to the physical computers, networks, operating systems, business programs (like word processors, spreadsheets and

the text editor I am writing this in) and workflow management systems like CRM and supply chain logistics. All of it exists to build a more efficient organisation and get some sort of competitive edge on the competition.

The other 1% of software systems are the unicorns of our industry. Held up as shining examples of what we somehow should aspire to with our business automation. Somehow, your system that handles the operation workflow for 10 people should be using the same tools and infrastructure technology that powers millions of transactions per second because that's the way that Google, Facebook, AWS, Netflix and so on do it.

This simply is a false idea. And in fact, a dangerous one that will probably cost you far more time and money and potentially bankrupt or seriously cripple the long term maintainability of your project.

You don't want countless levels of microservice abstractions or code that's hard to read because it's built for "webscale" and not mere mortals. What you want is something that a new developer can start on at 9am and could contribute meaningful code before she goes off to have her lunch.

So what would be the ideal characteristics of a software platform to build the vast majority of business software?

From my experience, there are 8 attributes and they are:

Fast to Build

Ruby on Rails is widely recognised as one of the fastest ways to rapidly iterate on a functional design to build an application and get it launched. The beauty of Ruby on Rails is that it allows you to create a Minimum Viable Product (MVP) that you can launch and continue to modify as you develop further features.

In other words, you don't have to throw away the money spent on an MVP, you can just continue to improve it.

As you are dealing with a simple stack that was one of the



Ruby on Rails pioneered a very simple 'everything has a place and a place for everything' approach to modern application frameworks. This results in a very rapid development cycle."

first frameworks to popularise the Model View Controller architecture, you get a simple, database connected framework that you can make any tool you need, fast.

But the biggest reason Rails apps are fast to build is that so many decisions have been made for the developer.

Ruby on Rails pioneered a very simple "everything has a place and a place for everything" approach to modern application frameworks. This results in a very rapid development cycle.

Easy to Maintain

The ‘fast to build’ aspect leads straight onto being easy to maintain. When you are attempting to debug an application, it sure is useful to know exactly where to look to handle the problem you are hitting. Of course, this doesn’t happen all the time, and sometimes it takes you longer to find what you need. But the fact is, that because of the design philosophy of Ruby on Rails, more often than not, you DO find what you are after relatively quickly.

Having worked on several other development frameworks, I have found this predominately not the case in the wider world. Each project has it’s own project layout decisions made by the team, with different class and abstractions for different layers of the framework and application logic.

The value of this is hard to quantify, but I had a new developer install a very non trivial multi tenanted Ruby on Rails application, on the other side of the world by just cloning the source code repository and following the readme, with no external support or questions able to be answered (I was asleep) and he got it done, and then fixed some things and wrote useful updates to the code to fix two issues I asked him to handle. All within the first few hours of ever seeing the app at all.

This is valuable. It gives new developers confidence that they can be productive and gives a product owner (AKA

the person paying the bills) confidence that they can add resources as needed or replace lost resources rapidly.

Well Supported

Ruby on Rails is incredibly well supported in terms of developer resource and software library depth. There are new versions coming out consistently with a very strong core team of developers pushing the boundaries of what is possible.

In 2016 Rails version 5 was released. This was a major version which expanded further on the core features available to get a new Rails application started with push notifications, faster page response times with Turbolinks and many more features. These include Yarn and Webpack among others that ensure the framework stays relevant with modern technologies and can move forward with new tools while still maintaining the convention over configuration approach.

2014 through to 2016, Ruby on Rails framework downloads (the first thing you need when creating a Rails application) has increased every year, 2014 with 12 million, 2015 with 15 million and 2016 with 24 million downloads, showing a growing community of users.

The 3rd party libraries built for Ruby and Ruby on Rails are called “gems” and these are hosted by the community on <https://rubygems.org/>. At the time of this publication, there were 130,746 gems published, over 111,371 open

source contributors and over 13 billion downloads.

These gems handle pretty much anything you could think of, from integrating with social networks or allowing you to create invoices in your favourite accounting system to providing pre-built, community vetted and reviewed authentication and authorization tooling.

In fact, in 2016, there were about 44 new gems released EVERY DAY with more than 300 new versions of these



Pretty much any major performance bottleneck in a Ruby on Rails application can be solved with some performance tuning."

gems released daily. On top of that the Ruby on Rails library was downloaded more than 24 million times with more than 92 million downloads overall and 463 thousand of them for the latest version. That's a lot of

people interested in the project.

How does this help? Well, as a product owner, when you want a new thing built, you can leverage a massive amount of 3rd party open source tooling, cutting development time by orders of magnitude and get that new feature released, fast, instead of having to build that whole feature from scratch.

Does this result in less cost in building apps? Perhaps not, after all, in my experience, if we build a part of an application faster than expected, it makes more

room for the development of additional features to be developed. However, it does mean that you get more bang (features) per buck out of your budget and more time can be spent polishing the application at the end of the project to make it really sing.

Testing Built In — Rapid Development

This is an often overlooked aspect of software development, though with recent movements such as Test Driven Development and Behaviour Driven Development it has been given more of a spotlight.

Ruby on Rails has a comprehensive, easy to use, automated test system built into the framework structure.

Automated testing means that as developers build the features, they also write an automated test that runs the application in such a way to ensure that feature works now, and into the future.

For example, say you require all users to have both a given and family name specified. The developer would write a test that tries to create a user without a given and family name and the test would check to make sure the system does not allow that to happen.

In the future, someone decides to alter this behaviour for another feature on the system, say for a newsletter sign-up where you only need an email. This might be months or years into the future, and the new developer has no idea about the requirement for given and family names when

the system was built, however, when she makes the change, the automated test suite will fail, pointing out that given and family names are required fields.

At this point, the developer can talk with the stakeholders and either fix the newsletter sign-up feature so it also requires given and family names, or change the test because it turns out that the business has changed and the names are no longer important, or find another way to handle newsletter sign-ups.

In any of those 3 options though, the result is the same. An automated test found an issue that if left un-confronted may have resulted in broken software being deployed to a production environment that could have prevented users from signing up correctly.

This simple example highlights how automated testing works. But the real benefit comes when future developers are employed to update your software.

You could imagine that after months of developing, the system would have hundreds, if not thousands of automated tests, checking all the features developed. Every time the code is changed, the tests are run, and if all the tests are green the developer has a high confidence that their new code hasn't broken anything critical.

This allows you to rapidly bring onboard new developers whenever needed. A new developer who gets your application source code and runs the tests and sees that there are a very significant number and that they all pass,

instantly has high confidence that the application is well maintained and well written.

Further, they can get cracking immediately on writing some new feature, confident that if they do something wrong, the test suite will show them what broke and they'll be able to rapidly resolve it – increasing productivity. The “best practices includes testing” approach in Ruby on Rails is a highly compelling reason alone to use the platform.

Improvable Performance

Ruby on Rails is built for high speed development first, and not so much high speed performance.

Witness Twitter that was built in Ruby on Rails, and after growing massively started having failures in being able to handle the traffic. But Twitter was dealing with more than quarter of a million users and in a write up after the fact held Ruby on Rails blameless¹ for their performance problems. There are very few business apps that would ever approach this scale.

Additionally, pretty much any major performance bottleneck in a Ruby on Rails application can be solved with some performance tuning. Modifying the database to return queries more quickly, keeping a copy of the results so they don't have to be recalculated every time, improving the way the web pages are loaded to take advantage of Content Delivery Networks and many other approaches.

My blog post about [Rescuing an Exploding Rails Application](#)² gives a good example of this. A client of ours had over 50,000 new users signing up per day and their Ruby on Rails application was having performance problems. After applying some of the performance tuning tools, we improved the application performance enough in a short time to handle the issue.

So, it's a rare case where Ruby on Rails cannot handle the performance demands required of it, and also a very rare case where a business application hits those performance limits and when they do, they can be solved almost all the time quite simply.

Also, in the exceptionally rare case that Ruby on Rails cannot handle the performance requirements directly, Ruby allows you to write and call highly performant C language functions directly, but in a decade of building Ruby on Rails applications, I have only seen this approach required a handful of times. Not bad.

Enjoyable to Use

Ruby is the computer programming language that Ruby on Rails is built with. Ruby's creator, Yukihiro Matsumoto said in an interview about the philosophy of Ruby that the goal of Ruby is to make developers happy while writing it. This might not seem like a big deal at first glance, but the benefits are far reaching, in fact, so much so that I built my entire company around developing in Ruby and Ruby on Rails.

Happy teams are productive teams. All of our developers get to program in a language that works with them, does things they expect, doesn't surprise them (too often) and provides them with a framework that they can produce code that they are proud of. Working with tools that work with you and you enjoy working with makes a big difference to how well you develop.

If you are a developer and get up every morning loathing having to code another line in the language your application is built in, you might not stay at the top of your game for long.

Conversely, if the programming language rewards your developers for writing well structured code and does what they expect it to do, they will most likely be more content with it and generate less bugs while writing it. They will probably stick around longer because the job is more rewarding and enjoyable. The impact a language like Ruby and a framework like Ruby on Rails has on increasing developer happiness is not to be discounted and actually should be a major reason for you to choose it.

Opinionated

Ruby on Rails is (just like this booklet!) quite opinionated. What do we mean by opinionated in the context of a web application development framework? Well, the software itself can't have an opinion obviously, however the people that built it and maintain it certainly do.

This opinion encourages developers to write code in a certain way, to put parts of the code base in certain places and to use certain programming idioms in the way they produce their code.

How does this benefit the person paying for the application to be produced? It means you get a certain standard across all Ruby on Rails applications. A Ruby on Rails developer can walk into a brand new Rails application they have never seen before and usually be able to find what is wrong with it in a quick and efficient manner because they know where to look as all Rails applications are structured the same way. This means the

“Pretty much any major performance bottleneck in a Ruby on Rails application can be solved with some performance tuning.”

new developer can go from never seeing the code base, to up and running very quickly.

It also means that when a developer needs to build a new feature, they don't have to think up a brand new decision about where to put that feature's code

in the application, instead, Ruby on Rails points out “That type of code belongs in this folder over here”.

You could imagine (and I have seen) other development frameworks not having this level of structure would result in code scattered wherever the developer thought at the time it would fit. These sorts of applications are an absolute horror to work on as it's very difficult to know what is where because there are no conventions.

I'm not saying that every Ruby on Rails application is perfectly structured with all the code in the right locations, but on average, from my experience, they are generally more consistent across different applications worked on by different developers. Again, this is a big plus for maintainability, which equates to future cost.

Cost Effective

Finally, this really is not a separate point, but the sum of the other seven prior points nicely wrapped up into something that makes clear business sense.

Building an app in Ruby on Rails is cost effective. It's cost effective in time. It's cost effective in resources. It's cost effective in hosting. And it's cost effective in maintenance.

Summary

So, Ruby on Rails provides you with a highly opinionated framework, that has testing built in, is easy to build and maintain, is widely supported throughout the world, has a massive pre-built open source library for integration to third party tools, can be optimised as needed when needed and on top of that helps make a happy development team, and overall is very well suited to build the vast majority of business critical front and back office systems out there in the most cost effective way. That's some pretty compelling reasons to use Ruby on Rails for your next project!

About reinteractive

reinteractive is the largest Ruby on Rails focused developer in the Asia Pacific region - we're not your average development shop.

With years of Ruby on Rails experience, we build stable, tested applications that improve your business bottom line through business process automation with quality code so you can be sure that your project is in good hands. We don't cut corners and we know what we are doing.

CEO & Founder, Mikel Lindsaar, is among the top Rails contributors world-wide. That same commitment to excellence extends to our entire team of experienced developers.

We bring our customers plans to life through well crafted and maintained software. We have a fully transparent, agile process keeping the client informed and in control of their project at all times.

We focus on Ruby on Rails because we believe it's the best choice for the web applications businesses need.

“We bring our customers plans to life through well crafted and maintained software. We have a fully transparent, agile process keeping the client informed and in control of their project at all times.”

reinteractive

Our services include:

- UX consulting & reviews
- Clickable interactive prototypes & designs
- Ruby on Rails design & development
- Mobile apps for iOS and Android
- API Integrations with most software
- Ruby on Rails upgrade support
- Ongoing server hosting and operations support with OpsCare®
- Ongoing code maintenance for your Rails application with CodeCare®

Find out more at:

www.reinteractive.com

info@reinteractive.com

Australia

Level 32, 101 Miller Street
North Sydney, NSW, 2060
(AUS) +61 2 8019 7252

USA

101 California Street, Suite 2710,
San Francisco, CA 94111
(USA) +1 415 365 9680



Confident Software

the secrets of successful software
development from one of the industry leaders.



MIKEL LINDSAAR

Making mistakes when building an application can cost you a lot of money.

Finding a solution to this requires first understanding what we are trying to achieve.

'Confident Software' is a book built from years of experience, exposing the fundamental laws for successful software projects that you can apply immediately.

**Get the secrets of successful
software development from one
of the industry leaders.**

Coming Soon!

**Register your interest at:
reinteractive.com/book/confident_software**



reinteractive

CONFIDENT SOFTWARE

www.reinteractive.com
info@reinteractive.com

Australia

Level 32, 101 Miller Street
North Sydney, NSW, 2060
(AUS) +61 2 8019 7252

USA

101 California Street, Suite 2710,
San Francisco, CA 94111
(USA) +1 415 365 9680



ISBN 978-0-6481026-0-1