# Using Truffle's ENS Integration

By Tyler Feickert, Blockchain Tools Engineer at Truffle

Among the new features released with Truffle 5.1 is support for Ethereum Name Service (ENS) name resolution and setting the resolver address for owned names!

ENS is a system that allows for the resolution of Ethereum addresses using human-readable names. Like DNS, ENS aims to simplify working with addresses and allowing us to work with names like "truffle.eth" in place of things that cumbersomely resemble something like"0x1234567890123456789012345678901234567890" . Are you sure you copied that address correctly? Ready to send your Ether there?

In ENS, a contract called a registry is deployed to a network. The ENS team has deployed a registry contract to Mainnet and several test networks. These are the registries that Truffle connects to by default if no registry address is set in the Truffle config.

An ENS registry contract contains a list of names and who owns them. Each name also points to a resolver contract if one is set. Addresses own ENS names and have the ability to set the resolver that a given name points to. The resolver is the contract responsible for returning an address for resolution. The idea is that the owner will have the ability to set the resolver contract to return the desired address.

For more detailed information on ENS, check out theENS website .

So we at Truffle think this project is pretty awesome and, as mentioned above, built an integration with it. Previously in your migrations, for example, you needed to manually deal with addresses. So maybe you wanted to send some tokens from your contract to another address. In your Truffle migration you might have the following code:

```
await

myTokenContract . sendTokens (

999 ,

"0x1234567890123456789012345678901234567890" ,

{

from :

"0x0987654321098765432109876543210987654321"

} );
```

It is difficult and tiring to deal with raw addresses like this.

Now in Truffle, if you turn on the ENS name resolution and you have a resolver set to your address of choice, you can do something like the following in place of the above:

```
await

myTokenContract . sendTokens (

999 ,

"truffle.eth" ,

{

from :

"my.account"

} );
```

Well, that seems much easier to read. Under the hood during the migrations, Truffle will connect to the on-chain registry and automatically resolve the addresses for both "truffle.eth" and "my.account". You now can use valid ENS names in place of addresses in your migrations! In other words, any place that an address is expected for an argument to a function call, you can instead provide the ENS name.

One other big piece of functionality for this ENS integration is the ability to set the resolution address for owned ENS addresses. You can do this usingdeployer.ens.setAddress in your migrations.

Suppose I control an address that owns the name "arnold.hagenchop.eth" and I want to set the address to "0x1234123412121234123412123412341212341234123412". In this case I would write something like the following in my migration:

```
const

myAddress

=

"0x1234123412123412341212341234121234123412" ; await

deployer . ens . setAddress (

"arnold.hagenchop.eth" ,

myAddress ,

{

from :

"0x8888888888888888888888888888888888888888"

} )
```

Now let's take a step back and look at what we did. So you can see that thesetAddress method has three arguments. The first one is the name that we'd like to set the resolver for. As stated above we want to set the resolver address for "arnold.hagenchop.eth". The second argument is the address to set for the given name's ENS resolution. The last one is an object that resembles a transaction parameter object. The important thing about this object is that it must have afrom property with the address that controls the name given as the first argument. This is important or the transaction will fail!

You can find some more information about registering ENS names from the ENS website .

One other useful convenience with this method is that you can also provide a Truffle contract object that has a deployed address as the second argument if you'd like. So the above code might look like the following instead:

```
const

MyContract

=

artifacts . require ( "MyContract" ); const

myContract

=

await

MyContract . deployed (); await

deployer . ens . setAddress (

"arnold.hagenchop.eth" ,

myContract ,

{

from :

"0x8888888888888888888888888888888888888888"

} )
```

In that case, "arnold.hagenchop.eth" will be set to the address at whichmyContract is deployed!

This new integration should just be the start of a much more robust system that allows Truffle projects to interact with ENS! In the future, we also hope to add other features such as reverse resolution (resolving a name from an address) and automatic name registration where possible. We think this will be a big step forward in making the Ethereum ecosystem more accessible as well as convenient to work with!

For more thorough information about this integration, see the Truffle docs . We have also created an example Truffle box that has some examples of using this new ENS integration alongside other new Truffle v5.1 features. We hope you find this feature as useful as we think it is! Happy Truffling!