

# How to Host a Project

Every project needs to be hosted on a server, somewhere. You can do this with permanent file storage or on a common server behind your own domain. Either way, the project is expecting to find specific information that it can use to show the visitor to your project.

## Background Thinking

The Selene Network is designed around the idea that humans will be providing value to other humans using the internet as a tool and the blockchain for settlement. Thus, projects that are presented to humans using the Selene Network protocols must be human friendly and allow product creators the flexibility to manage their offering in reasonable ways.

Yet, the ‘conversation’ is not one sided. When offering products, there is also someone that receives the product. The Selene Network works to make sure that the offered products conform to standards that the buyer can validate - at least to some minimum extent.

Ultimately, if an artist is offering one of a kind digital art that is delivered in the form of a JPG file, you would expect that the buyer would be able to hash that file and match the result to something on the blockchain. Making that match shows that the person that owns the JPG has an unaltered original.

Along this same line of thinking, if the artist’s website were to be shutdown so the project no longer shows up via the Selene Network, the owner of the digital art should still be able to reconstruct the original project to the point where they can show that the digital art they hold is an unaltered original.

## Hashing – used as a validation key

The process of hashing files is fundamental to how blockchains work and how they prove that the data their processing is unaltered.

When a project creator configures a Selene Network compatible smart contract using the `projectUpdate()` function, the third parameter to that function is a SHA1 hash of the project JSON file.

When this number is recorded to the blockchain, anyone can validate that the information in the project JSON file is unaltered. In other words, there are only two accounts (returned by the `titled()` and `owner()` functions) that can write to this smart contract. The value written as the project hash represents the validation key for the project.

Anyone should be able to download the project JSON file and hash it in order to generate the same key that is found in the project smart contract.

## Digging deeper regarding hashing

Once someone validates that the information contained within the project file is unaltered, they can trust the information stored in that file is correct.

This means that if hashes of other project files are stored in the project file, anyone that holds any one of these files will be able to validate that they, also, are unaltered.

This is the reason for the hashTable element of the project JSON file. The hashTable holds the SHA1 hash of all the important files in the project.

The projects that are produced by the creator of the Selene Network software include the files in the root of the project directory, the project directory files, the image directory files and key metadata directory files. And, when possible, a ZIP file as the same name as the project is provided on the server so anyone that wants to perform the validation of the project can.

### Caveats

The Selene Network project file is a step towards allowing the purchaser a path towards digital validation, but it is not perfect. It involves work to preserve enough information to allow for validation.

In other words, if a customer downloads a whitepaper.pdf file that is authored by a project creator which shows up in the project JSON file, the customer will be able to perform a SHA1 hash of the file to get the same value that would be reported in the project JSON file. Yet, if the project creator deletes their project from the server, there will be no JSON file use for validation.

To solve this problem, the customer that is keeping records of the project needs to save the project JSON file as well.

Because the Selene Network compatible smart contracts allow project creators the ability to update the server that they use to host the project, the project creator can also update information about the project on the same server. This gives them the ability to change the default hash that is reported for the project. Yet, the old record is still recorded on chain.

Thus, even though the project creator can update the hash used to validate the project (project changes), anyone that has authored material pre change has a path to showing that what they hold is authentic.

### Example

Project creator authors a project where it says “I will buy the winner a cheesecake” in the project whitepaper. A customer likes what he reads and performs some validation to secure their data. The customer downloads the whitepaper and project JSON file and hashes both and sees that the hashes validate that the customer holds unaltered data. The customer then buys the product knowing that they will win the cheesecake.

Later, the project author realizes that he can’t afford to buy a real cheesecake for his customers so he changes the whitepaper so that it reads “I will deliver to the winner an image of a cheesecake”. And then to make the project verifiable again, he updates the smart contract with the new project hash.

Because the author and customer are using Selene Network compatible smart contracts, the history of updates to the project are recorded on chain. The customer can query the smart contract for all the recorded hashes. In order to prove that the customer holds a

legitimate authored whitepaper is to show that it matches any of the hashes that the author produced.

Thus, in this case, the customer can show that the author changed the terms of the agreement and the blockchain will show when that change occurred.

## Pre-host Development

It has proven helpful to the developer of the Selene Network to outline the project on his development machine following the exact layout that is requested by the Selene Network.

To give you an idea of this process, the first thing that is created is the name of the project which is used to create a subdirectory on the development machine. For this document, we'll assume that it's called 'myproj\_1' Then, in that folder three subdirectories are created: images, project and metadata (and maybe even private). (Note that all file names should be lower case!)

Any images used by the project are placed in the images directory. All the files used to build the smart contract are placed in the project directory and the token specific metadata files are placed in the metadata directory. If there is content that will be token-gated, it should be placed in the private directory and maybe even in a subdirectory therein.

It's important to only place the files that will be used by the project in this directory so that it is easy to simply send the folder to a ZIP file that can be uploaded to the server for hosting. Yet, if there is content in the private directory that is audio or video, you might want to exclude that from the ZIP file that gets uploaded and upload that separate.

## Review the Smart Contract

All smart contracts will have different file requirements. It's important to double check what those requirements are before proceeding.

For this document, we're expecting the user is creating a document based upon the drmaudiov1 Selene Network compatible smart contract. This contract will require that you create a unique image that can be used to identify the project which should have the same name as the project and be located in the images directory. You can create either a JPG, PNG or GIF. And, of course, the author will present some type of data that will be token-gated.

## Building the Project JSON file

The project JSON file should have the same name as the folder that it is in. Following the example from above, if the folder is called 'myproj\_v1', the project file that you create will be 'myproj\_v1.json' which will be placed in the 'myproj\_v1' folder.

Likewise, the image that describes the project will also most likely want to be named something like 'myproj\_v1.jpg' and it will be placed in the images directory.

### *Tip*

It is recommended that you install a JSON file viewer in your browser. Make sure you install one that allows you to toggle between raw and formatted.

### *Tip*

It is recommended that you download an already functioning project JSON file and modify it to suit your own needs. Because JSON is just text, any text editor will allow you to edit into what you require.

### [The Selene Network tools directory](#)

To assist with the development of the project JSON file, the Selene Network core code can process your file to guide you as you make changes.

Here is an example of how you would review your project file:

<https://amorstyle.com/dsn/tools/?project=https://amorstyle.com/nfts/drmaudiov1/drmaudiov1.json>

Note that there are three sections of interest: Initialization data, project file data and core contract data.

Initialization data is the information that is needed by the projectUpdate() routine that must be set after contract launch.

The core contract data is the smart contract addresses of the main contracts that are needed to track Selene Network participation.

The project file data is the information that is parsed to show the project in the Selene Network core code.

And with this example, there is an audio section. The audio section is parsed by the custom code to show 9 audio tracks. The information provided by each object in the JSON file will be shown to the visitor, thus it's important to get the details correct. This is also the place where you describe the subfolder and file name of the file on the server that will be downloaded to the token holder.

Also note that the hashes for each of the files should show up in the hashtable.

### [Download a project JSON file to modify](#)

This document is going to assume that you have downloaded an example project JSON file and renamed it for your own use and that you can view it in your browser.

### **Element descriptions**

#### *name*

short name. Title of the project. It is a required element.

#### *description*

Single paragraph. Keep is short. This is also a required element.

#### *external\_url*

URL to webpage or website where more information can be found. It's worth noting that information authored at the location pointed to by this URL is not hashed into the project

file. This means that the location this points to can be dynamic and updated by the author of the project without affecting the project's resulting hash.

This metadata element is optional, but it is expected to be on the same server as the overall project.

*whitepaper*

URL to PDF file that is the whitepaper of the project. The hash of the whitepaper is expected to be in the hashTable of the project file. Thus, if the author changes the whitepaper, the author will want to update the project with a new project hash (see the function projectUpdate()).

Note that this element is optional, but if it is provided, it is expected to be located on the same server as the overall project.

*image*

URL to image (jpg, png or gif) that represents the project.

This element is required. The image is expected to be located on the same server as the project.

*artist*

Just a short name. Optional element.

*creator*

Just a short name. Optional element.

*collection*

Just a short name. Optional element.

*home*

Subdirectory location if there is custom code used by the project. If the project doesn't use custom code, it should not provide the home location directory.

Note that the tools project code reads this entry and builds the home location. The code echos this built URL to the ui so you can validate that it points to the correct custom code.

Because the core code builds the home path, it is expected that the custom code lives on the same server as the project and that it is a php file.

*type*

The Selene Network uses this keyword to provide the reel, film and music notes overlays on the NFT/DATs. It is a string and, if used, should be either 'reel', 'film' or 'music'.

*hashTable*

This is a list of the important files stored on the server that are associated with the project. The idea here is that when you publish some digital content, you provide the SHA1 hash so the visitor can be assured that the resource has not be tampered with.

By providing the SHA1 hash of the object, anyone can download the resource and validate that they have the correct copy.

The files listed in the hashTable should match the files that are included in the project ZIP file. This makes it so that anyone can download your project ZIP file in order to validate the agreement behind the token that the customer purchases.

Before outlining the hashTable, you need to know how to generate a SHA1 hash.

#### [How to create a SHA1 hash](#)

It is expected that one of the items in the hashTable is the line:

```
"tool": "certutil -hashfile filename SHA1"
```

Every Windows system comes with the certification tool that allows you to generate a hash of a file. This line describes how to use the tool in a command window.

You'll want to produce a SHA1 hash of each file that you include in your project.

#### [Layout of hashTable](#)

The 'locations' section of the project JSON file is designed to represent where a file is located in the directory structure of the project along with providing the name and hash.

All the files that are included in the root of your project should be included in the root section.

Any directory that you create for files should have it's own section and mimic the images, project and metadata sections.

#### [Token Metadata JSON](#)

The project file (above) is used to describe a project. In a similar way, each token needs to provide its own unique metadata.

The unique metadata that is referenced by the Selene Network samples is in the metadata directory. When the smart contract builds the reference to the token's JSON information, it uses the two parameters to the projectUpdate() function and then appends 'metadata' and the project name again in order to reference the unique elements for the individual token.

All metadata files that will be referenced by the smart contract need to be checked to validate that they work as expected. To assist with this process, the Selene Network core coded tools processing also allows you to process a metadata file as a 'token.' Here is an example:

<https://amorstyle.com/dsn/tools/?token=https://amorstyle.com/nfts/drmaudiov1/metadata/drmaudiov1.json>

As with the project file, the token file must contain the name, description and image elements. The other elements are optional and follow the same rules as with the project file.

The metadata associated with an individual token will not have a hashTable, but projects may introduce attributes which (may) require custom coding. Developing Attributes are more complex and are not (generally) used by the Selene Network, thus its not covered here.

Also note that individual token files should consider supporting the ‘type’ field. The Selene Network wallet functionality uses this keyword when showing the clickable functionality of the NFT/DAT.

## Ultimate Goal

The idea of mimicking the layout structure needed by the Selene Network on your local machine means that you can simply ZIP the folder contents and upload it to your server. Simply remove the old files and extract the new ones.

It’s at this point that you can finish the setup of the smart contract that you’ve launched on chain or read over the smart contract guide document that is provided in this sample.

## To Review

Determine what smart contract you want to use or build one.

Once it is ready, build the hosting data that this document describes and author it to your server.

When you have confidence that you have put together a working solution, launch the smart contract on chain and perform the post launch configuration.

It is at this point that the Selene Network should be able to display your project. On any server that has the Selene Network core code installed, add ‘?contract=your\_contract\_address’ to the URL and confirm the project displays as you would expect.

## Corrections

If you find a spelling error or typo that needs to be corrected, make the appropriate change and update the project. For instance, you change the image that represents the project. You would generate a new hash for the project JSON file and upload the information. After modifying the hosting data, you’ll need to write a new hash to the smart contract using the projectUpdate() routine.

Note that the Selene Network installs will query the smart contracts in order to get the recorded hash that should match what the server provides. If it matches, the Selene Network code will display a blue ribbon. If it doesn’t, a yellow ribbon.

## Disclaimer

As with any code, there could be bugs. Use this code at your own risk. If you find a bug, please fix it, and then let people know what the fix is. Also, this project depends upon other third-party tools and utilities, which could have or develop blocking bugs. Thus, use this code at your own risk.