**Developing Plutus Applications - FAQ**

1. **Developer Onboarding and Developer Environment Setup**

**Where can I learn how to develop Plutus Applications?**

We are launching a developer portal soon to support onboarding, education and networking needs of developers. The portal will help developers learn how to develop Plutus Applications in the Cardano network, collaborate with each other and share projects across the community. Read more about it [here](https://forum.cardano.org/t/a-first-look-at-the-cardano-developer-portal/40544). You can access the Developer Portal here.

**Are there any reference implementations of example use cases?**

Developers will have access to reference implementations of example use cases in Plutus and will be able to run these examples in an emulated Cardano environment in the [Plutus Playground](https://alpha.plutus.iohkdev.io/).

Like the whole of Cardano, the Plutus Platform is being developed as an open source project and there is a wealth of documentation available in the [Plutus github repo](https://github.com/input-output-hk/plutus).

**Can I develop Plutus Applications in Haskell? Are there any constraints or limitations compared to standard Haskell?**

Yes, developers can write Plutus Applications in Haskell. Most of the standard Haskell needed to author Plutus Applications is supported. For more details about Haskell support in Plutus please see [here](https://github.com/input-output-hk/plutus/blob/master/plutus-tx/README.md).

**How can I set up a local development environment to develop Plutus Applications?**

Developers can consult the [Plutus Platform project](https://github.com/input-output-hk/plutus/blob/master/README.adoc) on github to find source code and documentation on how to set up a local environment for developing Plutus Applications. Some setup options are described in [this guide](https://github.com/input-output-hk/plutus/blob/master/example/README.md).

1. **Plutus Applications in the Context of the Cardano multi-asset ledger**

**How do Plutus Applications interact with the Cardano ledger?**

The Cardano ledger is based on an extension of the Unspent Transaction Output (UTXO) model pioneered by Bitcoin, which we call the Extended UTXO Model (EUTXO). In a nutshell, these extensions allow for custom user code, called a validator (that can be developed using the Plutus Application Framework) to run on-chain as part of validating transactions on the Cardano multi-asset ledger. In addition, the Plutus Application Framework allows developers to construct transactions that will trigger the validator to run during transaction validation. More details about the Extended UTXO Model can be found in [this paper](https://hydra.iohk.io/build/4847959/download/1/extended-utxo-specification.pdf).

**Which information/data from the ledger can be accessed from a Plutus Application?**

An on-chain component of a Plutus Application (called a **validator**) can access data included in the transaction which it is validating, input passed to it by the off-chain component constructing the transaction (called a **redeemer**), and some state information that can be passed between subsequent executions. More details in the EUTXO paper.

**Can I use user-issued assets within my Plutus Application?**

Yes, user-issued assets can be used within Plutus Applications in the same way as Ada.

**How can I construct transactions to progress the state of a Plutus Application?**

The Plutus Haskell SDK allows you to easily construct transactions, which is typically done within the off-chain component of a Plutus Application. These transactions will have one or more transaction outputs locked by a validator script, requiring that validator to be run in order to spend such outputs. Remember that the validator is an on-chain component of the Plutus Application, typically implemented as a state machine and, as a result, the state of the Plutus Applications will be progressed when it is invoked.

**How do Plutus Applications interact with a wallet to get permissions to access the funds kept in the wallet?**

Transactions constructed by Plutus Applications need to be signed and submitted to the Cardano ledger by the user's wallet. This process will result in asking for user’s approval each time before a transaction is submitted. Read more about how [here](https://github.com/input-output-hk/plutus/blob/master/doc/tutorials/basic-validators.rst).

Alternatively, you can still construct your transactions manually (using the command line interface of a Cardano node) by following the process described in [this document](https://docs.cardano.org/projects/cardano-node/en/latest/reference/building-and-signing-tx.html).

1. **Some Details on Design, Interoperability, and Available Infrastructure**

**Where can I learn more about the design choices behind the Plutus Platform and their implications on Plutus Applications development**?

The design of the Plutus Platform, including how it relates to EUTXO ledger model and Plutus Core scripting is outlined in [The Plutus Platform Technical Report](https://hydra.iohk.io/build/4848074/download/1/plutus.pdf).

High level architecture of the Plutus Platform can be viewed [here](https://input-output.atlassian.net/wiki/spaces/GD/pages/1684176997/Plutus%2BArchitecture).

**How can I securely access off-chain data from a validator script of a Plutus Application? How are oracles supported with Plutus?**

At launch of the Plutus Platform, we will provide mechanisms to make ***signed*** off-chain data from a ***trusted, centralized*** source accessible in a secure way from a validator script of the Plutus Application (i.e. from the component of the Plutus Application that is executed on-chain). This mechanism can be used to access data like price feeds and exchange rates from sources like centralized crypto exchanges. More information on how to do it will be available shortly.

**How can Plutus Applications interact with other Plutus Applications? Can they share common code?**

An off-chain component of the Plutus Application (handling transaction construction and user interaction logic) can simply expose endpoints that can be accessed from any other application running on the same machine.

Running an on-chain component, i.e. a validator of a Plutus Application can result in constructing a new transaction with one or more outputs associated with a different validator. This takes advantage of the EUTXO model to allow validators to interact with each other via transaction outputs.