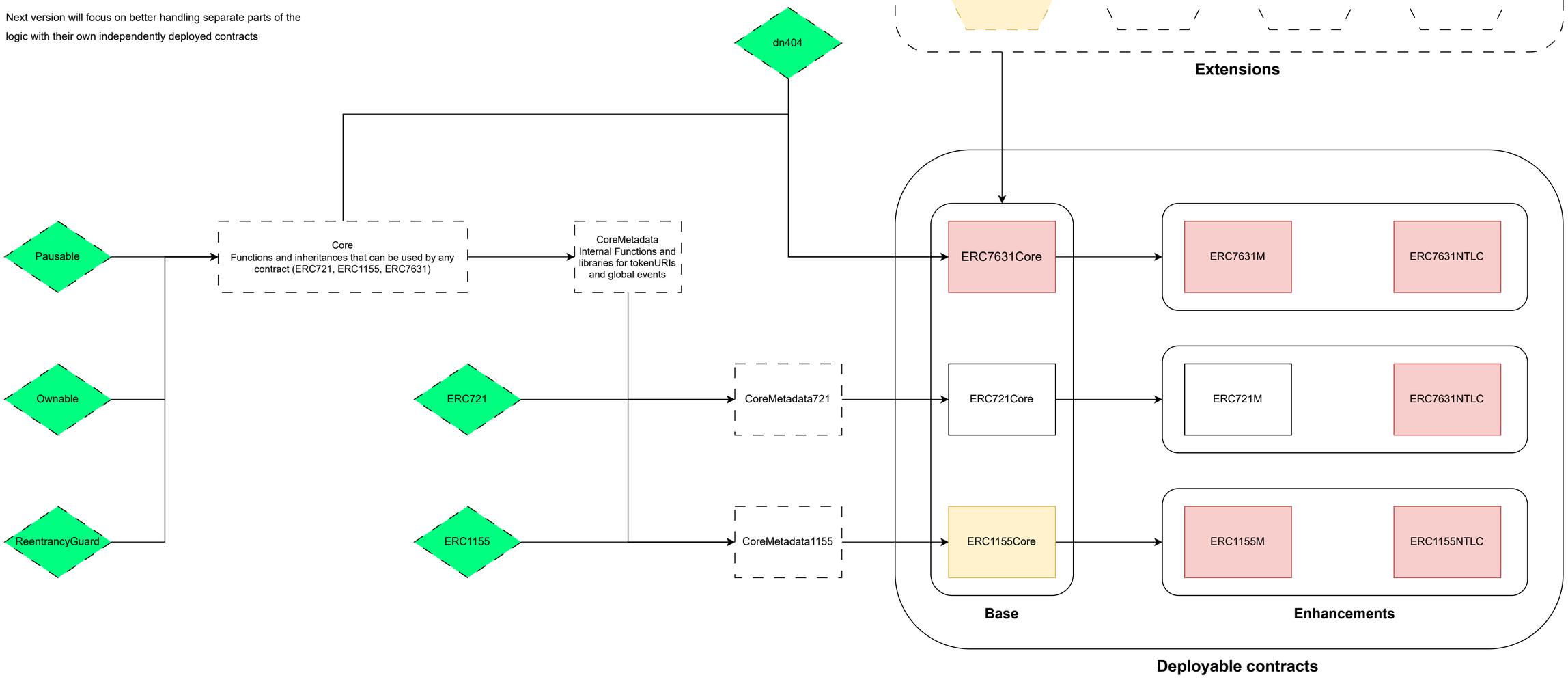
CRATE NFT Contract V1

This page describes the various parts of the nft contract that gets deployed by the factories and interacts with external balance managers such as ERC721M AlignmentVault or NTLC Treasury.

This version is already deemed transitional.

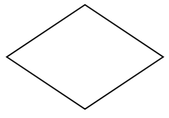
Next version will focus on better handling separate parts of the logic with their own independently deployed contracts



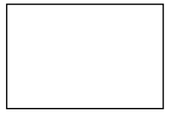
Legend

Short explanation of what symbols and colors mean

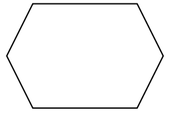
Shapes



Rhombus: External primitive, most likely unusable by itself



Rectangle: In-house contract, a required dependency of a given Base Contract



Hexagon: An extension to a base contract, unusable by itself

Shape Lines

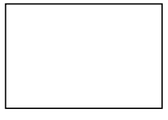
Dashed Line: Contract unusable by itself, needs to be a dependency of some other contract which needs to implement the necessary missing functions

Solid Line: Contract usable by itself and therefore deployable

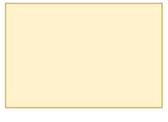
Colors

Ready + Tested + Audited

Ready + Tested



Ready



Work in progress

Not Started

A note about factories

Due to the modular nature of the contracts it's counterproductive to pre-deploy all valid combinations on any given chain.

That does not necessarily mean that a offchain-aided deployment is unsafe.

Contract addresses for all given combinations can be precomputed through the usage of an Immutable CREATE2 factory at a fixed address.

Addresses can then be stored by the factory/factories in order to ensure the intent of the factory deployer to support the contract.

On a hypothetical frontend we will then check what address we will need to clone given the particular set of contracts and extensions indicated and, if not available, proceed to deploy and verify the contract at the established address.

Further plasticity could be achieved through passing raw calldata for constructor and initializer, alongside the combination address, in order to jumpstart the contract.

While exploring alternative solutions like external modules will make us less reliant on this method, it's hard to say if we will be able to completely avoid this or similar solutions.