


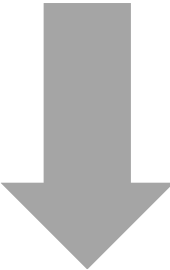
Token Taxonomy Framework (TTF)

Moving Tokens Forward

What its for:

- 
- **Educate** – take a step back and CLEARLY define a token in non-technical and cross industry terms. Using real world, everyday analogies so ANYONE can understand them using properties and behaviors to describe and define them.
 - **Establish** a common set of terms and definitions that can be used by business and technical participants to speak the same language
 - **Create** - implementation neutral token definitions (artifacts) with clear requirements that developers can follow, and standards validate following a composition framework
 - **Define** meta-data using the TTF syntax and grammar to be able to generate visual representations of classifications, used by modelling tools to view and create token definitions mapped to the taxonomy and can also support taxonomy to implementation or source code mapping

It is **NOT**:

- 
- Blockchain specific
 - A legal framework – but a common ground
 - Complete or comprehensive – requires and encourages collaboration

Introduction

Token Designer

The interface is titled "New Token-1" and features a central canvas with a dialog box that reads: "Drag a Token Base on this canvas to get started." with "OK" and "Cancel" buttons.

Token Bases:

- Fractional Fungible
- Whole Fungible
- Whole Non-Fungible
- Fractional Non-Fungible
- Singleton
- Unique Fractional

Property Sets:

- SKU
- File
- cusip
- BOL
- Attestation
- ...

Token Formula:

[Empty text box]

Behaviors:

- attestable
- burnable
- compliant
- delegable
- singleton
- encumberable
- holdable
- issuable
- logable
- mintable
- non-sub-dividable
- non-transferable
- overdraftable
- pausable
- roles
- subdividable
- transferable
- ...

Behavior Groups:

- Supply Control
- Financable
- Insurable

Token Designer

Token Bases



Fractional Fungible



Whole Fungible



Whole Non-Fungible



Fractional Non-Fungible



Singleton



Unique Fractional

Property Sets



SKU



File



cusip



BOL



Attestation



...

Token Formula

$$\tau_F\{S\}$$


New Token-1



Behaviors



attestable



burnable



compliant



delegable



singleton



encumberable



holdable



issuable



logable



mintable



non-sub-dividable



non-transferable



overdraftable



passable



roles



subdividable



transferable



...

Behavior Groups



Supply Control



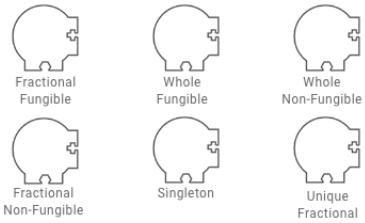
Financable



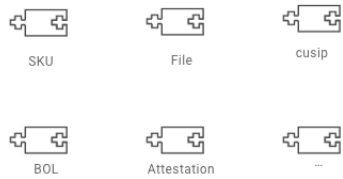
Insurable

Token Designer

Token Bases



Property Sets



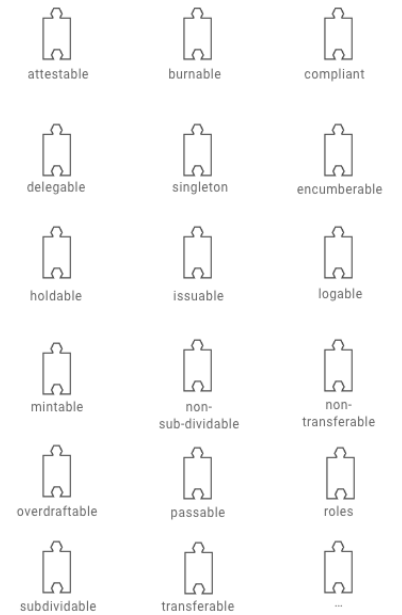
Token Formula

$[\tau_F\{s\} + \phi File + \phi BOL]$

New Token-1

Singleton Token File BOL singleton

Behaviors

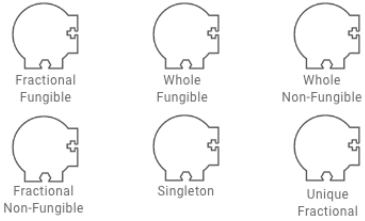


Behavior Groups

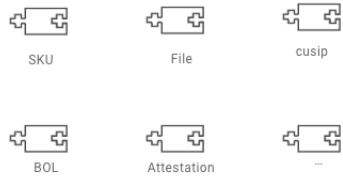


Token Designer

Token Bases



Property Sets



Token Formula

$[\tau_F\{s, t, a, l\} + \phi File + \phi BOL]$

New Token-1

Singleton Token

File BOL

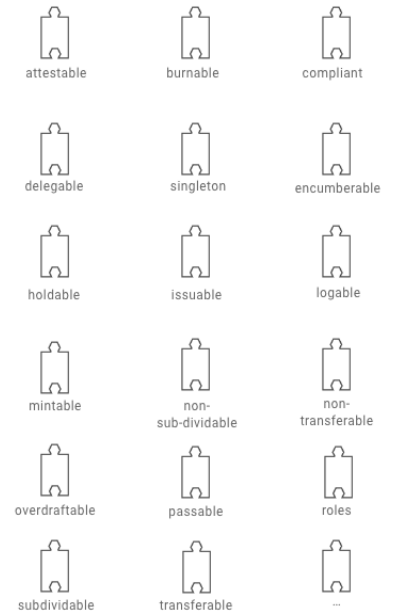
singleton

transferable

attestable

logable

Behaviors

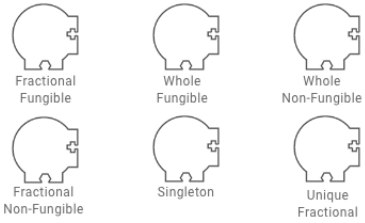


Behavior Groups

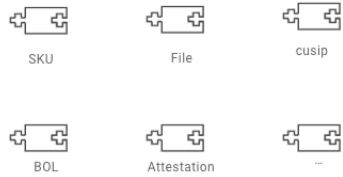


Token Designer

Token Bases



Property Sets



Token Formula

$[\tau_F\{s, t, a, l\} + \phi File + \phi BOL]$

New Token-1

Singleton Token

File BOL

singleton

transferable

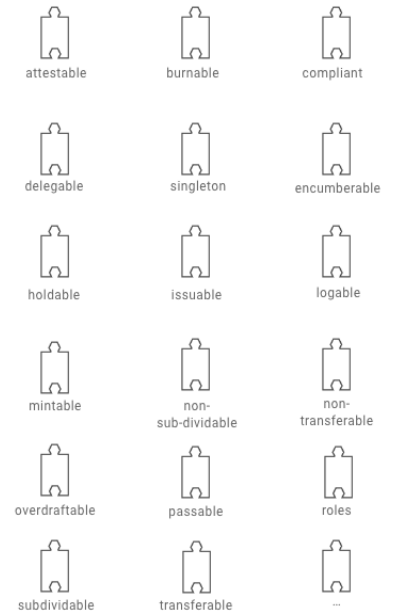
attestable

logable

Save As: Bill of Laden

Save Cancel

Behaviors



Behavior Groups



Token Designer

Token Bases



Fractional Fungible



Whole Fungible



Whole Non-Fungible



Fractional Non-Fungible



Singleton



Unique Fractional

Property Sets



SKU



File



cusip



BOL



Attestation



-

Token Formula

$$\tau_F\{d\}$$

New Token-1

Fractional Fungible

subdivisible

Decimal Places ✓

Example
888.88

2

0

1

2

3

Token Template Values

Base Properties

Decimals

2

Quantity

1,000,000

Base Properties Behaviors Property

Token Designer

The interface is divided into several sections:

- Token Bases:** A grid of six icons representing different token types: Fractional Fungible, Whole Fungible, Whole Non-Fungible, Fractional Non-Fungible, Singleton, and Unique Fractional.
- Property Sets:** A grid of six icons representing different property sets: SKU, File, cusip, BOL, Attestation, and an empty set.
- Token Formula:** A text box containing the formula $[\tau_F\{d\} + SKU]$.
- New Token-1:** A central workspace showing a diagram of a token. The token is composed of three parts: a 'Fractional Fungible' part, a 'SKU' part, and a 'subdivisible' part. A callout box for the 'SKU Properties' is open, showing:
 - Max Length: 16
 - Regular Expression: `/^[a-z0-9_-]{3,16}$/`
 - Instructions: Use this to set field requirements like starting sequence or location of special characters like a `^`.
- Token Template Values:** A panel on the right showing the configuration for the 'SKU' property set:
 - Max Length: 16
 - Regular Expression: `/^[a-z0-9_-]{3,16}$/`

Token Designer

The interface is divided into several sections:

- Token Bases:** A grid of six token base icons: Fractional Fungible, Whole Fungible, Whole Non-Fungible, Fractional Non-Fungible, Singleton, and Unique Fractional.
- Property Sets:** A grid of six property set icons: SKU, File, cusip, BOL, Attestation, and --.
- Token Formula:** A box containing the formula $[\tau_F\{d\} + SKU]$.
- New Token-1 Canvas:** A central workspace showing a token design. It features a 'Fractional Fungible' base and an attached 'SKU' property set. A tooltip for the 'SKU' property set is displayed, containing:
 - SKU** (checked)
 - Description: A token class that implements this property set will have a SKU property or field with a Read/Query and Set control.
 - Example Retail** (eye icon)
 - Text: SKU number for the item being inventoried which will apply to all tokens in this class.
- Right Panel:** A detailed view of the 'Fractional Fungible' token base, including a description and a business example.

Fractional Fungible

Token Base

Fractional Fungible tokens have interchangeable value with each other, where any owned sum of them from a class has the same value as another owned sum from the same class. Similar to physical cash money, a crypto currency is an example of a fungible token that

Business Example: Physical Money or Cash

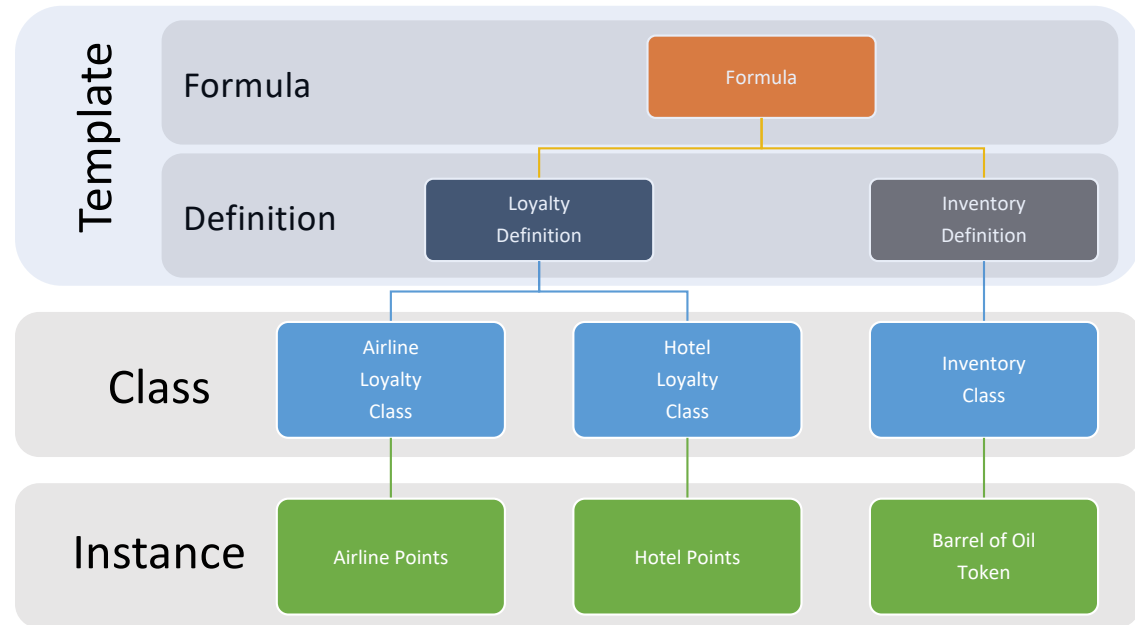
Cash, or fiat money, is freely accepted between parties and can have varying denominations. Money has a face value, on a coin or bill, and can be summed together to represent higher value. It can be subdivided, making change, and consolidated from many smaller denominations to larger ones and still have the same value.

Base Properties

Behaviors

Behavior Groups

High Level Terms



The framework establishes a set of terms for tokens like template formula, definition, class and instance

Token Template = Formula + Definition

An instance belongs to a class which is created from a template

Artifacts are definitions of a token part or complete token definition, template

The framework consists of the base set of artifacts and any new artifact contributed

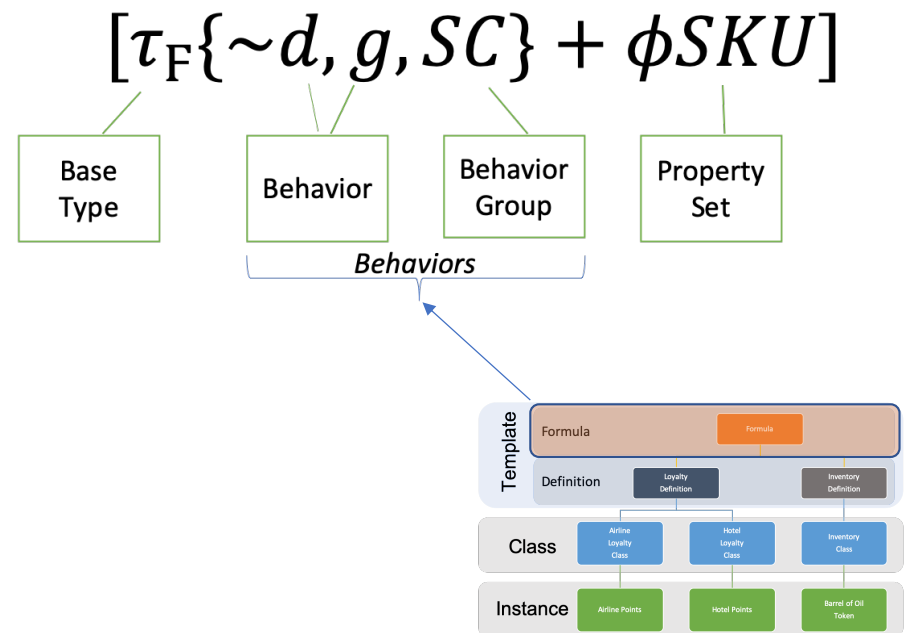
Baking with Artifacts, Templates & Instances



- A Token Class is like a Cake, it is baked from a recipe, i.e. a Template
- A slice of this cake is like a Token Instance
- A cake is made of ingredients like milk, flour, sugar, etc.
- The cake recipe specifies how much of each ingredient to use, when and how long to bake it.
- In the TTF:
 - Ingredients are Artifacts
 - Recipe is a Template
 - A Cake is the Class made from the Template
 - A slice is an instance of the Class

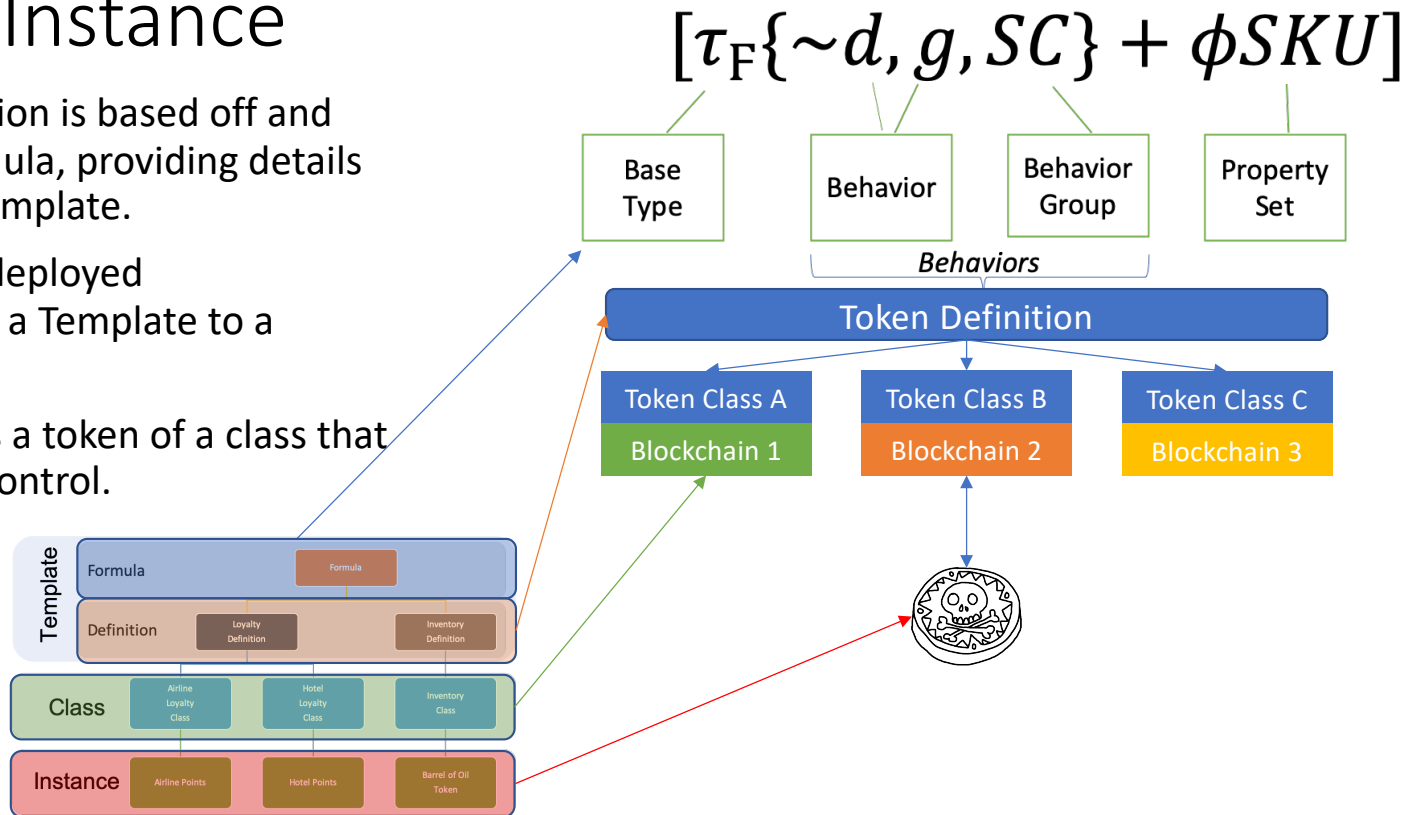
Template Formula

- Collection of components from the TTF
- Represented by a Formula of Symbols
- Symbols link to individual artifacts
- Formulas are used to classify templates



Template Definition, Class & Instance

- A Template Definition is based off and paired with a Formula, providing details to complete the Template.
- A Token Class is a deployed implementation of a Template to a blockchain.
- A Token Instance is a token of a class that you can own and control.



Categories: Base Token, Property Sets, Behaviors & Behavior Groups



Tokens are comprised of properties and behaviors that are interacted with using standard control messages



All Tokens will have a common set of base token properties including simple name/value pair generic properties.



Tokens differentiate themselves based on their behaviors and property sets.



The Taxonomy defines behaviors and properties that are the building blocks of tokens. Some behaviors are already known, the taxonomy just defines them in a standard way and organizes them.



Controls are actions described as messages that are used to invoke a behavior and represent properties.

Artifacts

- GitHub based
- Artifact Categories using File System Folder/Directory structure
- Extensible
- ArtifactSymbol contains a formula symbol as well as a unique identifier or id which is used to reference the artifact.

Written in common language so anyone can understand the artifact description and what it does.

Metadata for: (hidden from users)

- Tokens
- Behavior
- Behavior Groups
- Property Sets
- Token Templates

An artifact is a collection of files placed in a single folder based on the artifact type.

```
{
  "artifact": {
    "type": "BEHAVIOR",
    "name": "Delegable",
    "aliases": [
      "To Delegate"
    ],
    "artifactSymbol": {
      "visualSymbol": "g",
      "toolingSymbol": "g"
    },
    "controlUri": "../../../artifacts/behaviors/delegable/delegable.proto",
    "artifactDefinition": {
      "businessDescription": "A token class that implements this behavior will support",
      "businessExample": "",
      "analogies": [
        {
          "name": "Analogy 1",

```


Base Token Types & Symbols

τ_F Common Fungible

$\tau_{F'}$ Unique Fungible

Interchangeable having the same value with other tokens of the same class

- Physical money – $\tau_{F'}$
- Loyalty points - τ_F

τ_N Non-fungible

$\tau_{N'}$ Unique Non-fungible

Different values and should not be interchangeable.

- Property title
- Art token

$\tau_N(\tau_F)$

$\tau_F(\tau_N)$ Hybrids

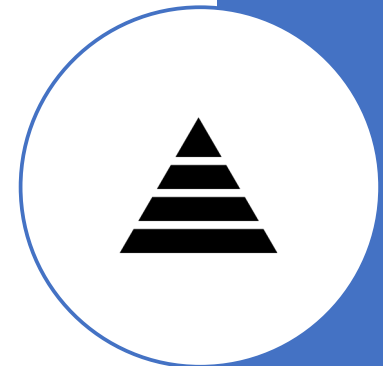
- Theater ticket – non-fungible base – Play name, show date + classes of non-fungible tokens for theater sections.

- Artifact group of tokens owned by other tokens and represent them as a single token.

Classification

A Token Classification has 5 Variables:

1. Token Type: Fungible or Non-Fungible – the fundamental difference between token types.
2. Token Unit: Fractional, Whole or Singleton – Quantity and subdivision restrictions.
3. Value Type: Intrinsic or Reference – the asset type the token represents.
4. Representation Type: Common or Unique – how balances and property value settings are stored.
5. Template Type: Single or Hybrid – does the token have any child tokens.



Classification Details Beyond Fungible & Non-fungible

Token Unit: Fractional, Whole or Singleton

- Fractional – you can make change like a \$1 dollar bill can be broken into 4 .25¢ coins, but you cannot subdivide past 2 decimal places or have a .249999¢.
- Whole – no subdivision allowed just whole numbers
- Singleton – no subdivision and a quantity of 1

Value Type: Intrinsic or Reference

- Intrinsic value is where the digital token itself is valuable, a crypto currency
- Reference value is where the token represents a physical item like a car or house, or 'stored elsewhere' digital item like a photo, scanned document or bank balance.

Representation Type: Common or Unique

- Common tokens are balances on a single distributed ledger, tokens do not have individual identities. Like the balance in a checking account.
- Unique tokens have their own identities, usually called an unspent transaction output or UTXO, that can have individual properties like a serial number. Like physical money in your pocket, each bill has a unique serial number.

Template Type: Single or Hybrid

- A hybrid token has a single parent token of a classification and can have many child tokens, that 'belong' or are 'controlled' by the parent. But like real children, hybrids can have unique abilities to model almost any business use case.
- A single token does not have any children.

Properties

- **Behavioral property** – represent a value that is determined through logic or calculation and not set directly.
- **Non-behavioral property** - represent a value where the value can be set directly without computation by the token implementation.
- **A Property-Set artifact** can contain the definition of a single or multiple properties like a **SKU** property or **Issuing Organization**.
- Property sets have a ϕ prefix and a capital letter or acronym that is unique for the taxonomy. For example, ϕSKU could be used for the `SKU` property set.
- A property defined in a set artifact is explicit.
 - Behavioral properties may have a query control message but not define a control message. Setting the value for the property is done by its behavior.
 - Non-behavioral properties will have control messages to query and set values.



Behaviors



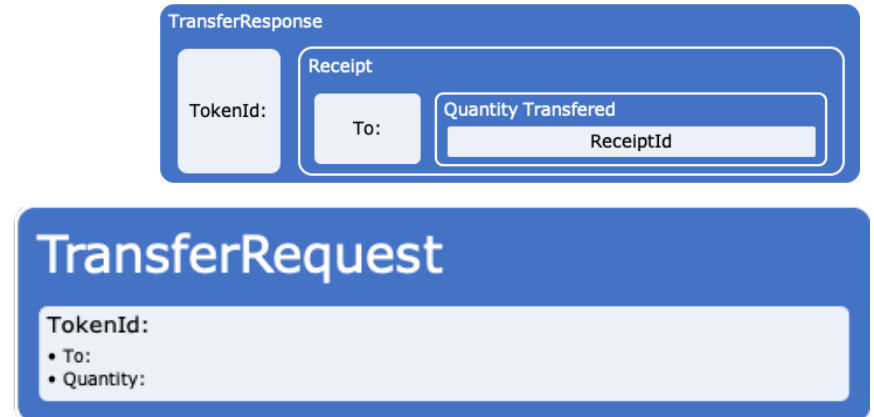
Behavior – describes a capability or restriction. I.e. Transferable or Non-transferable.



Behavior Property – a property or data element that is required for a behavior. I.e. Sub-dividable requires a decimals property.

Control Messages

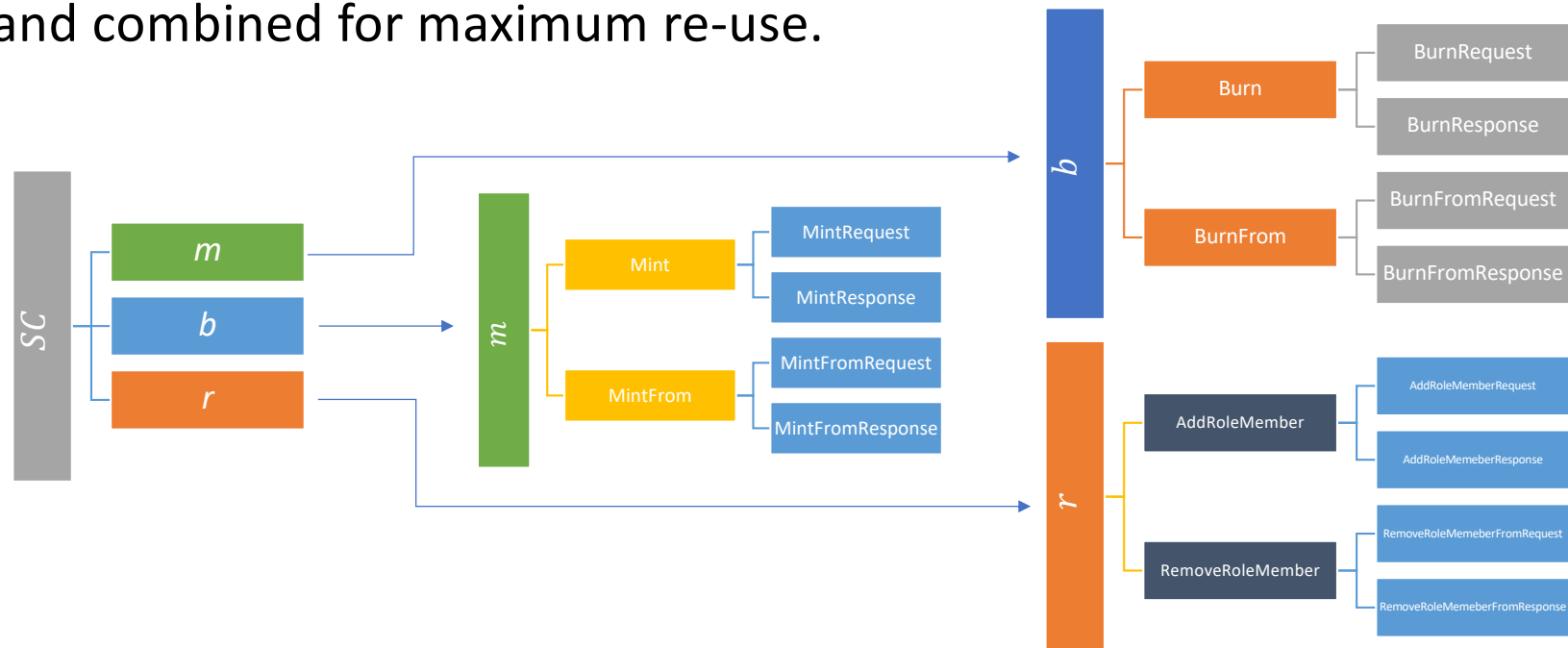
- Property-Sets and Behaviors have Control Messages that describe how to get or set the property or invoke a behavior.
- Messages usually named starting with a verb like Get or Add
- Control messages are paired together with a Request and a Response.
 - Request messages describe what property to get or behavior to invoke along with any parameters needed for the Control. For example, a TransferRequest message will need to have To and Amount parameters to successfully invoke the Transferable behavior.
 - A Response message describes what the Requestor is expecting in return from their response. I.e. a receipt, query result, etc.



Common Behaviors

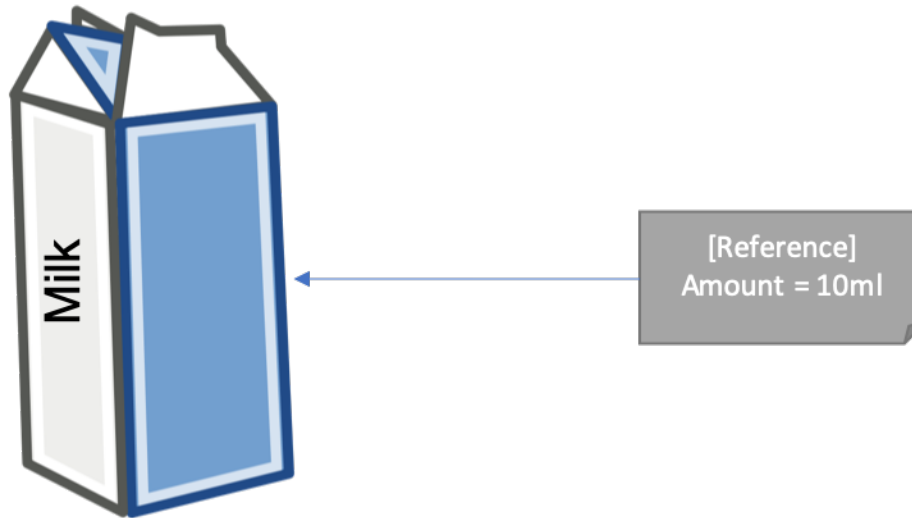


- Behavior Groups are a collection behaviors that are frequently used together.
- Encourages Decomposition of lowest level behaviors being defined and combined for maximum re-use.



Behavior Groups

Artifact Reference

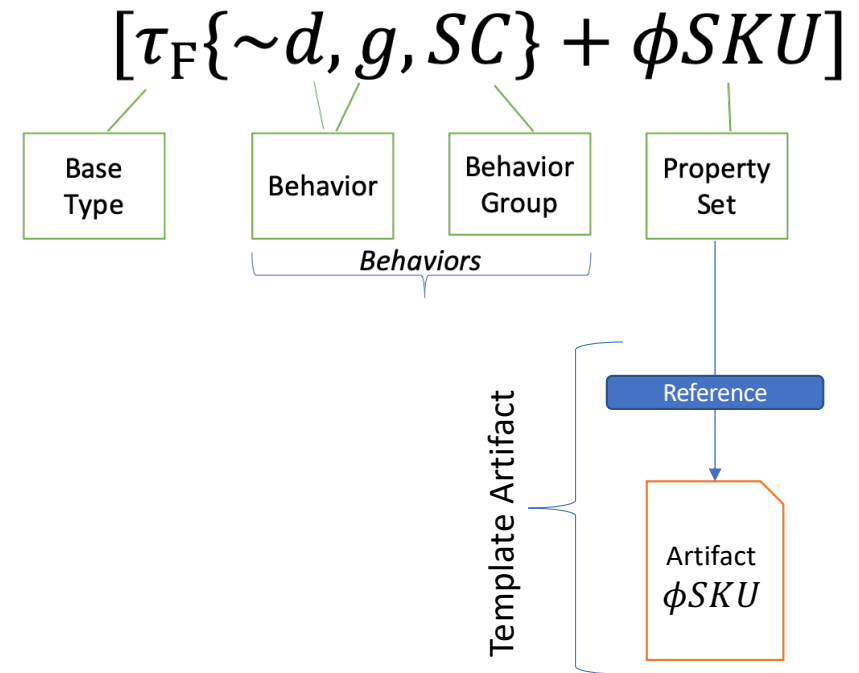


- An Artifact is an ingredient, like milk or flour, and has a placeholder for the amount needed of the ingredient, but not the actual value.
- An Artifact Reference is where the value is set for the Artifact's use.

Template Formula

Setting Context for Artifacts working together

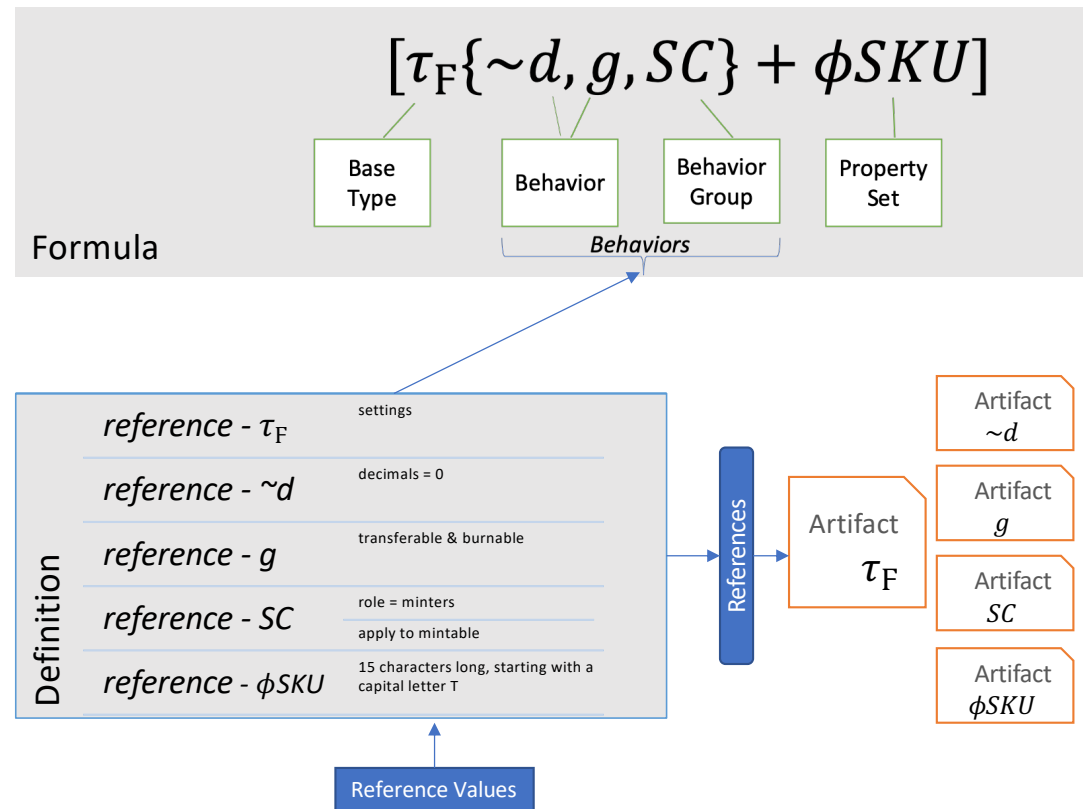
- Artifacts are composed into a formula using references.
- Grammar rules will provide the first level of component compatibility.
- Formulas represent a branch on the Taxonomy Hierarchy Tree.



Template Definition

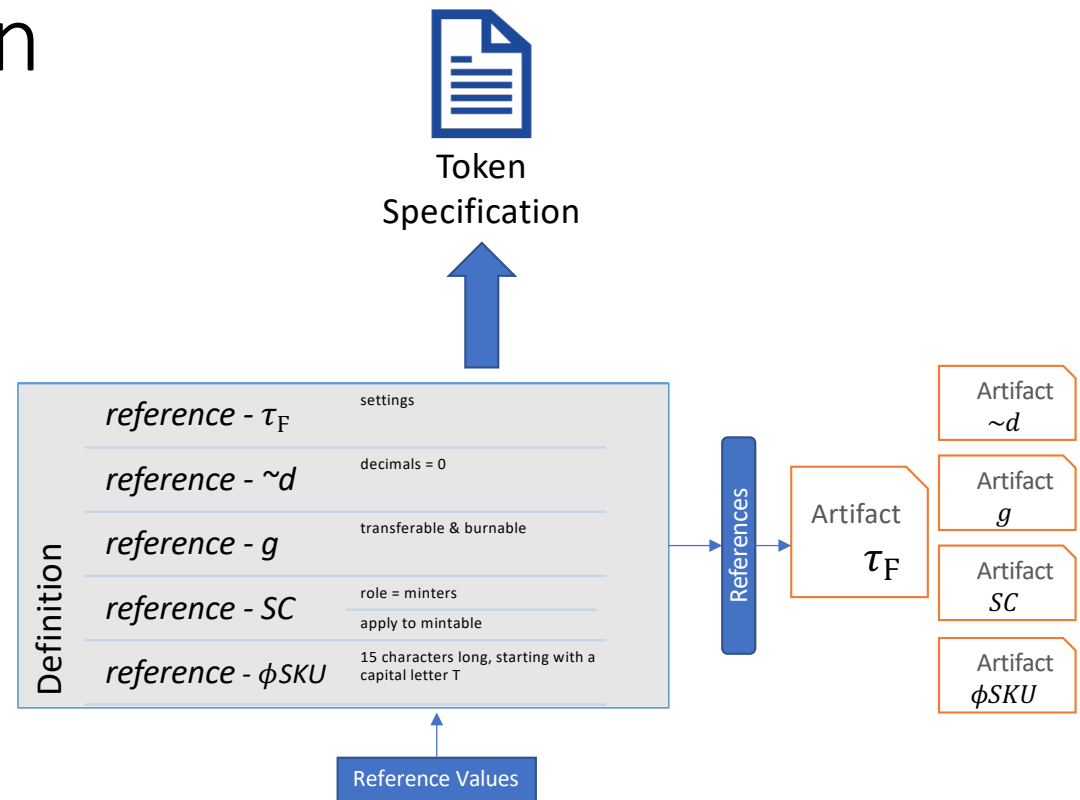
Filling in the details...

- Individual Taxonomy Artifacts are defined in relative isolation to facilitate reuse.
- A single artifact may list a dependency on another artifact but leave details like property values or settings as TBD.
- When artifacts are composed together into a Definition, the TDB values can be specified, setting values can be set or recommended for these artifacts in the context of the template.
- A Token Template is where Definition is a paired with its Formula
- The Definition Id is the also the Token Template and Token Specification Id.



Token Specification

- A token template is the combination of its formula + definition.
- From a template definition a specification is generated.
- The TTF generates the specification upon request from the Template Definition Id



Taxonomy Grammar

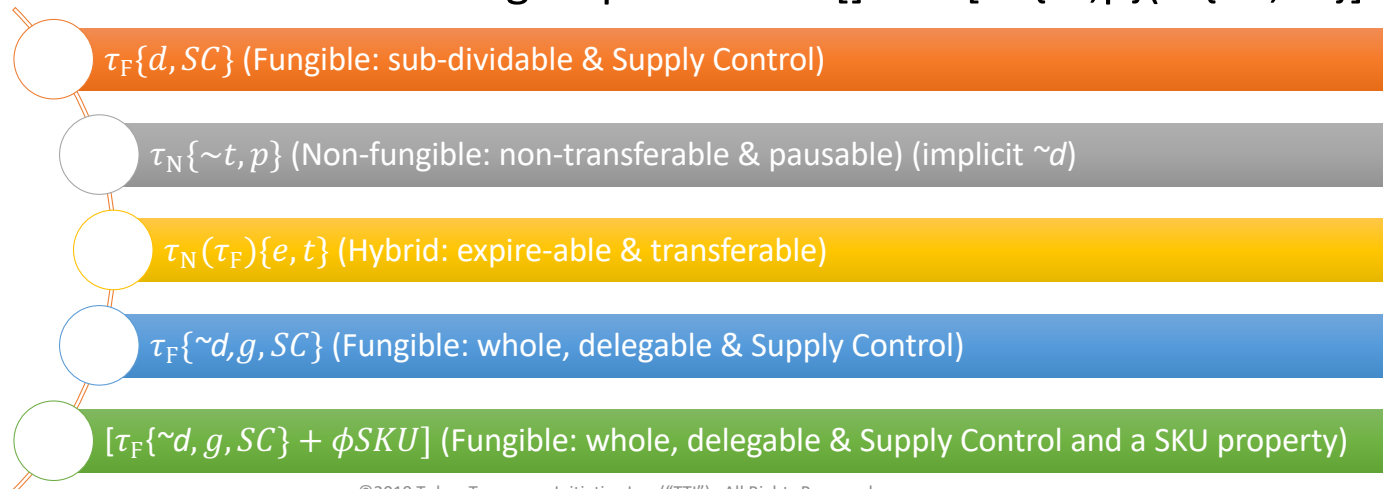
- Grammar defines how to construct a token or behavior group formula that is recorded as metadata in the artifact for the respective token, behavior or group.
- The grammar has a visual and tooling format where the tooling does not include presentation characters for *italics*, Greek, super or subscript, etc.

Token Base Type	Visual Formula	Tooling Format
Fungible	τ_F or $\tau_{F'}$	tF or tF'
Non-fungible	τ_N or $\tau_{N'}$	tN or tN'
Hybrid – classes in (,)		
Non-fungible with a class of fungibles	$\tau_N(\tau_F)$	tN(tF)
Fungible with a class of non-fungibles	$\tau_F(\tau_N)$	tF(tN)
Fungible with a class of non-fungibles and fungibles	$\tau_F(\tau_N, \tau_F)$	tF(tN,tF)
Non-fungible with a class of fungibles and non-fungibles	$\tau_N(\tau_F, \tau_N)$	tN(tF,tN)

Taxonomy Grammar

- Behavior: single *italic* lower-case letter or letters that are unique
- Behavior Group: upper-case letter or letters that are unique with behavior formula encased in {,} Supply Control: SC{*m,b,r*}
- Whole Token Formulas start with the base token type, followed by a collection of behaviors and groups in {,}
- Property Sets used to create a node or leaf from a base formula are added by wrapping the the formula in brackets [] and then adding + the property-set(s) using braces if there a multiple sets required.
- Hybrid children with formulas are grouped within []: i.e. [tN{~t,p}(tF{~d,SC})]

Formulas



Behavior grammar/rules



Some behaviors can apply to a token's class, like **delegable** (*g*) and can **influence** behaviors added to the definition of the token like transfer and burn.



Some behaviors have a natural opposite or a need to indicate a lack or disabled behavior. These will have the same symbol, where the opposite is indicated by a prefix of \sim .



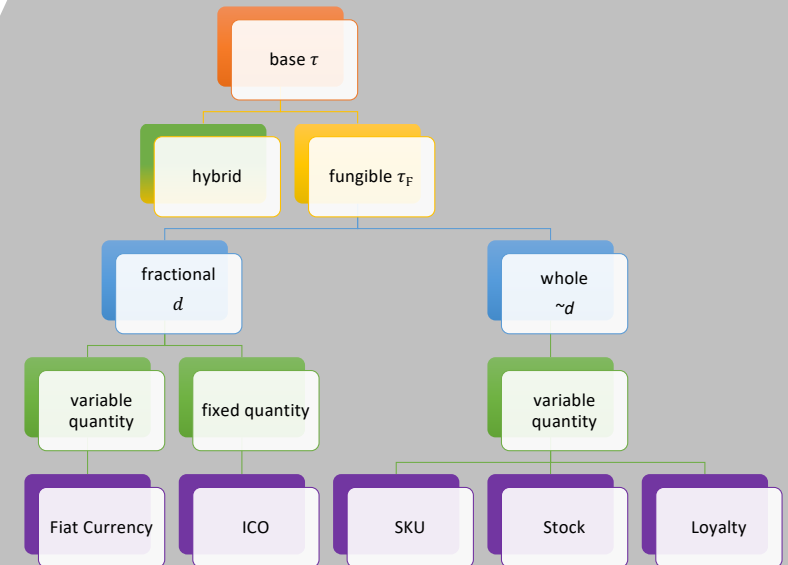
Some behaviors are incompatible with others.



Some behaviors are implicit based on the base token. I.e. transferable, is implicit for fungible tokens.

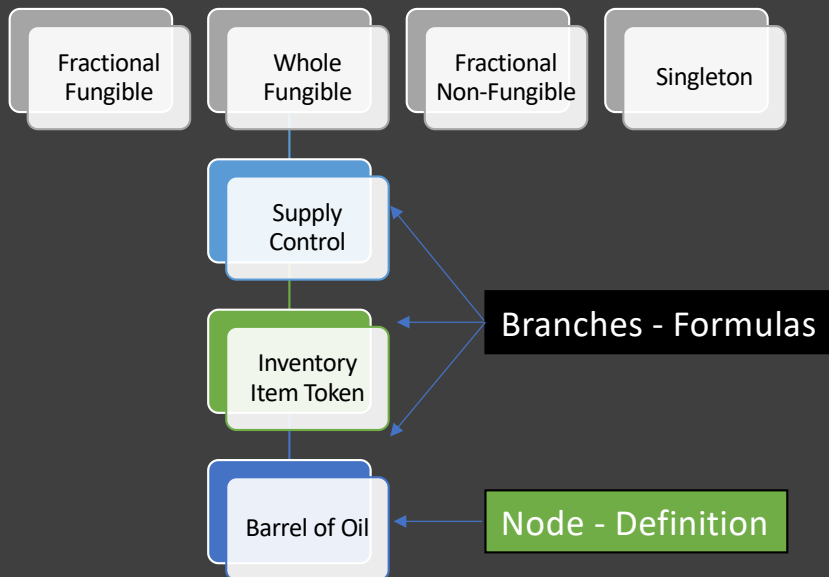
Branch Classification

- Common root Base - τ
- Classification views:
 - 3 branches: Fungible τ_F and Non-Fungible τ_N and Hybrid
 - Branches and views can be created using Template formulas like Fractional Fungibles, Whole Fungibles, Fractional Non-Fungibles or Singletons, etc.
- Branch views are created from template references and behaviors.
- Template Formulas are branches
- A Definition is a node or leaf
- Token Template = Formula + Definition



Classification Hierarchy

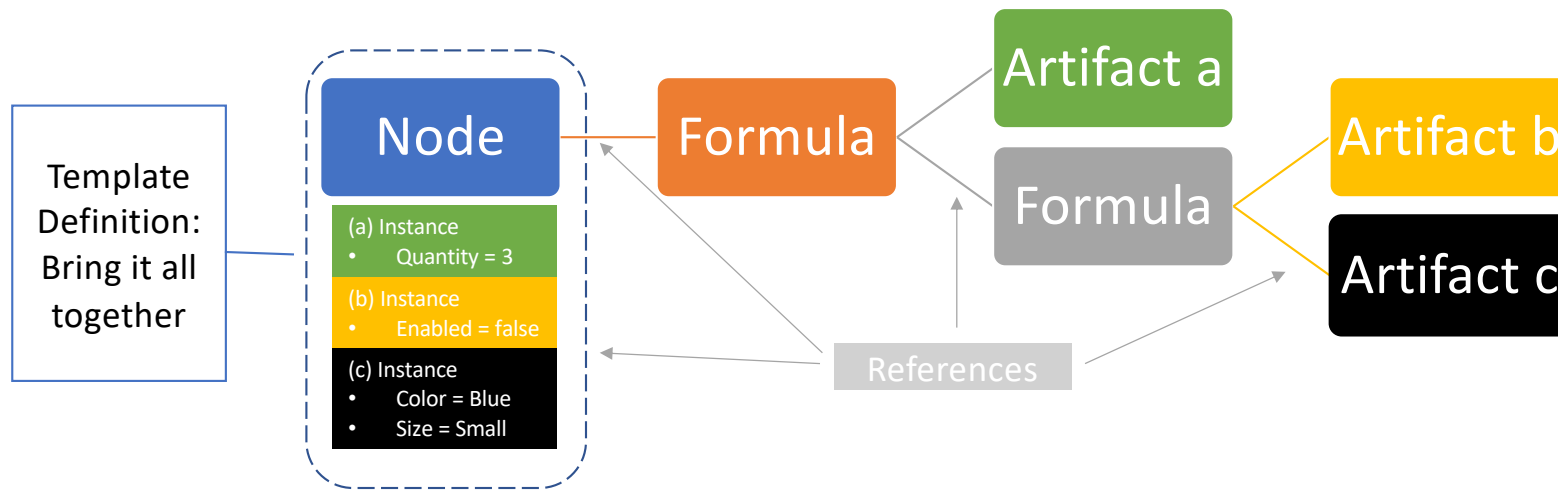
Formula Branch	Visual Format	Tooling Format	Token Name
$\tau_F\{\sim d, SC\}$			Whole Fungible with Supply Control
	$[\tau_F\{\sim d, SC\}] + \phi_{SKU}$	$[tf\{\sim d, SC\}] + phSKU$	Inventory Item
	$[\tau_F\{\sim d, SC\}] + \phi_{CU}$	$[tf\{\sim d, SC\}] + phCU$	Bond



- A Token Node or Leaf represents an end-point or destination in classification hierarchy.
- A TTF Template Formula represents a branch, which can reference another branch to reuse and add to its formula.
- For example, you can have multiple templates that share the same base formula $\tau_F\{\sim d, SC\}$ or Whole Fungible Token with Supply Control, but differ based on the property-sets they require. These would be different branches each referring to the same parent.
- A Template Definition is a Leaf or Node on the branch for the formula it is based from.

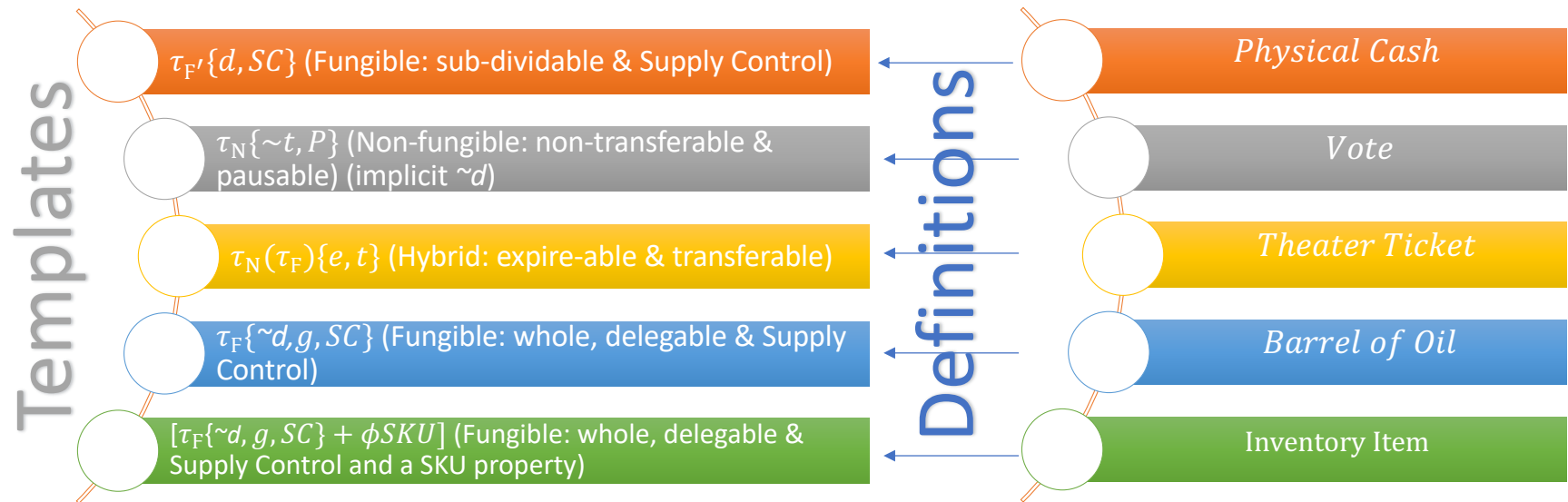
Token Template = Formula + Definition

- A Token Template Formula is a Branch
- Definitions based off a Formula are leaf or nodes supplying the values for each Artifact Reference in the Formula.



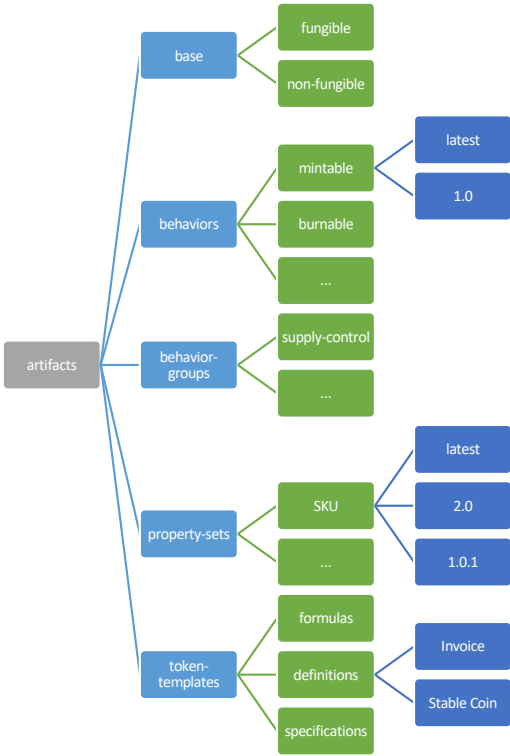
Token Template

- Definitions must have a Formula
- Formulas can be the source of many different definitions
- Multiple definitions using the same template should have significantly different Artifact Reference values



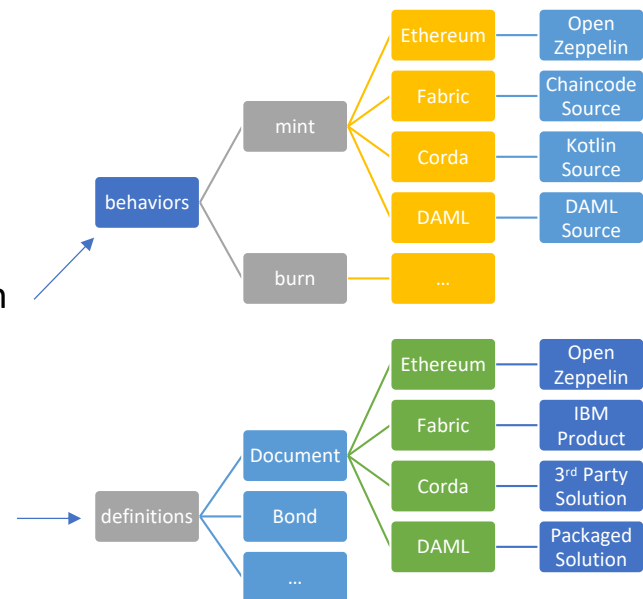
Artifact Hierarchical File Structure

- Taxonomy Framework Repo stores artifacts in folder structures by base type, behaviors, behavior-groups, property-sets and tokens.
- Version specific folders contain all the artifact files for that version. Latest always contains the latest version regardless of version number.



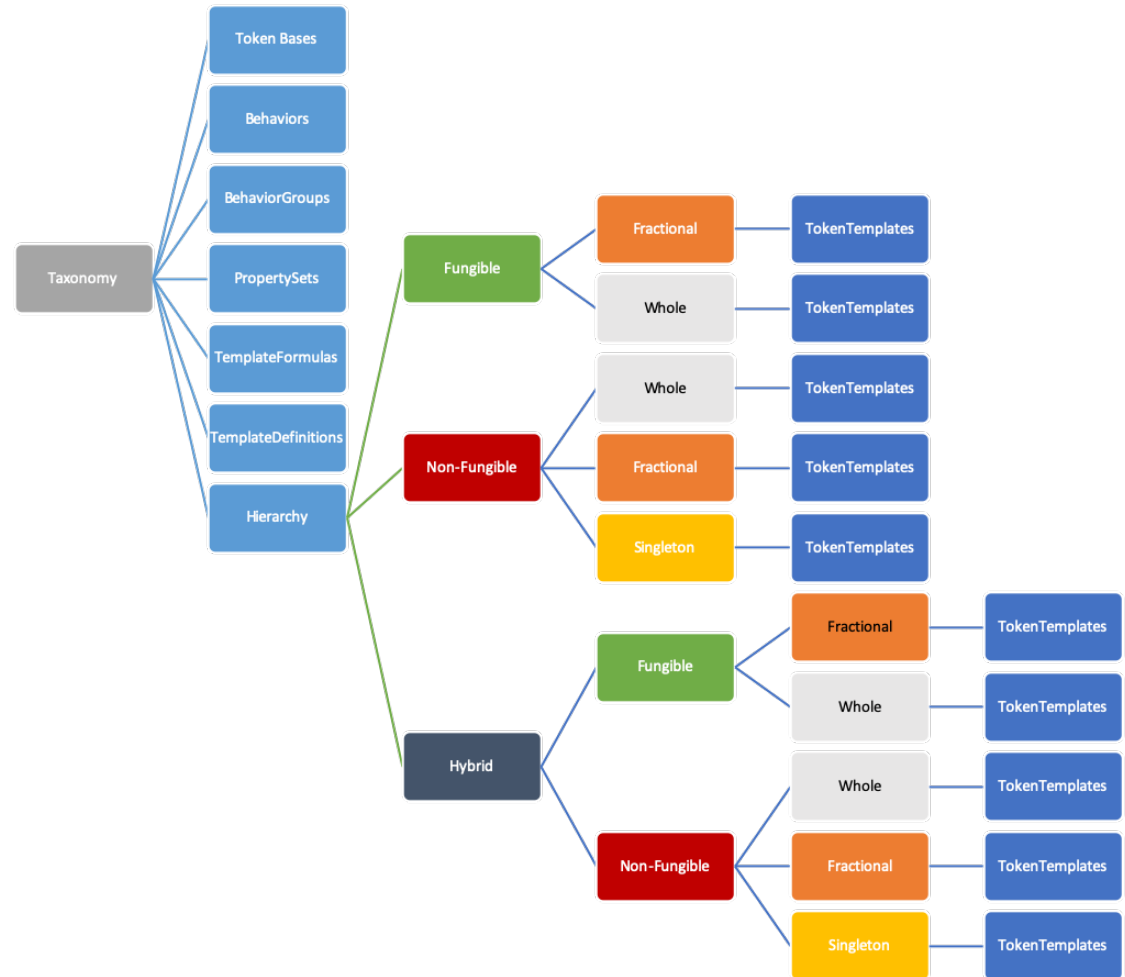
Tooling with Artifacts and Metadata

- Artifact metadata supports mapping to:
 - Source code
 - Finished implementation
 - Resource (i.e. regulatory framework or guidance)
- Map for taxonomy behavior or property-set to platform specific code. (Solidity, Chaincode, Kotlin, DAML)
- Map for token template to a specific implementation. (complete open source, commercial solution, etc.)
- Which Map entry to use can be specified in an Artifact Reference



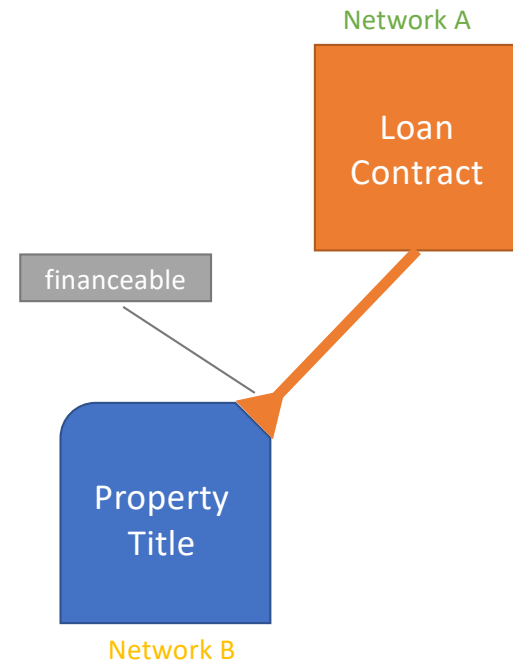
Taxonomy Model

- The Taxonomy is represented as a structured model with all the artifacts in the TTF
- From these tools can build design palates as well as visual representations of the artifacts

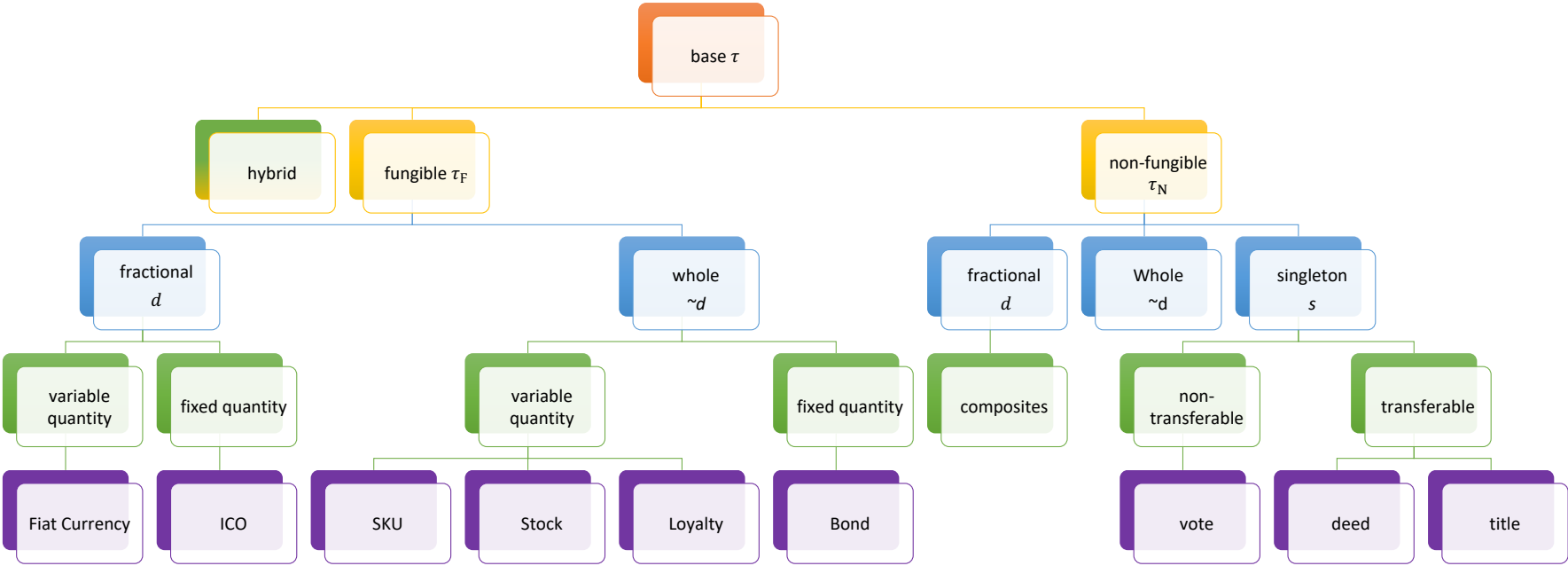


Contract Interfaces

- A Template Formula represents the logical token interface.
- Internal behaviors apply only to the token itself and are self invoking.
- External Behaviors provide standard contract interfaces across platforms and consortiums. These behaviors are often “end point” properties invoked by an external contract or token.

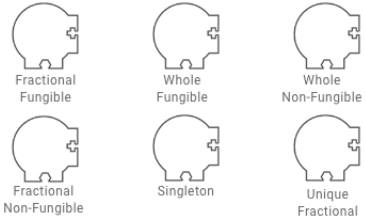


Classification Hierarchy Example

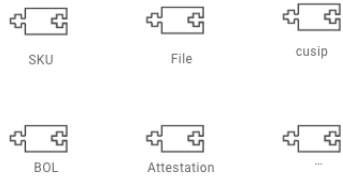


Token Designer

Token Bases



Property Sets



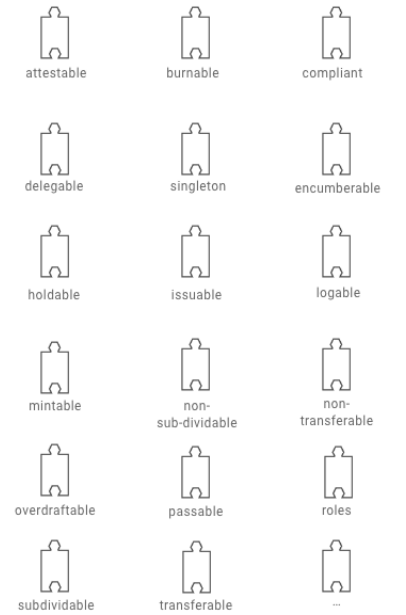
Token Formula

$[\tau_F\{s, t, a, l\} + \phi File + \phi BOL]$

New Token-1

Singleton Token File BOL
singleton
transferable
attestable
logable

Behaviors

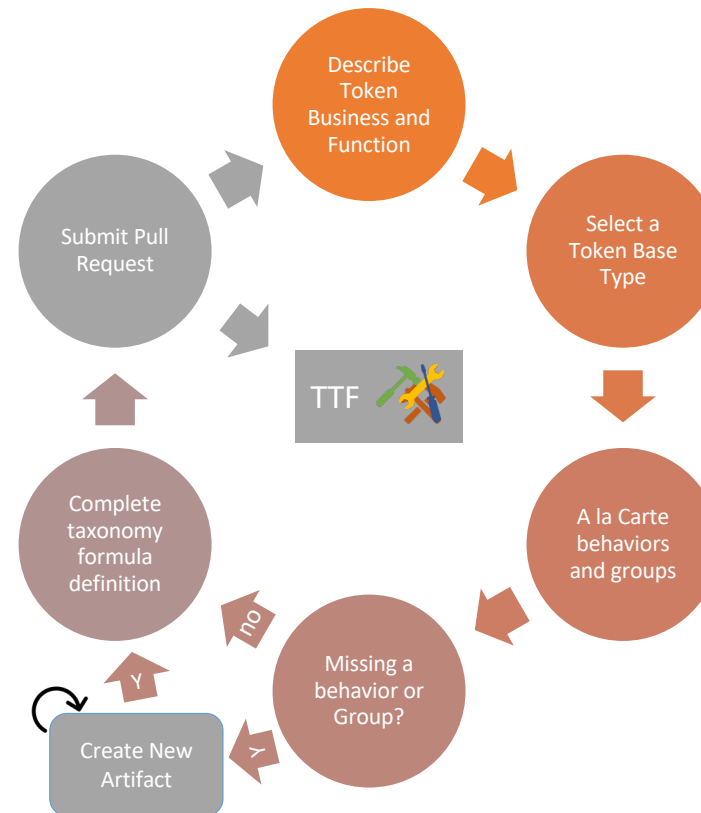


Behavior Groups



The TTF is designed for collaboration and extension:

- TTF initially has only the most common generic artifacts.
- Workshops will contribute most artifacts and token templates.
- Subject matter experts in vertical businesses or innovation will define and contribute artifacts in their area of expertise.
- Those artifacts are then available for reuse and refinement by all.



Workshop Process

Where does the TTF end & implementation begin?

- The TTF has extensions, called a Map, allowing each artifact to reference code, implementation or reference materials that are external to the TTF.
- For example, you may have a reference for a Token Template Definition that maps to:
 - Complete source code for a platform and language (i.e. HLF/Chaincode/Go - > url)
 - A finished solution or implementation (i.e. a link to a website or marketplace offering)
 - Regulatory guidance (i.e. link to a regulatory agency report)





Questions

<http://tokentaxonomy.org>

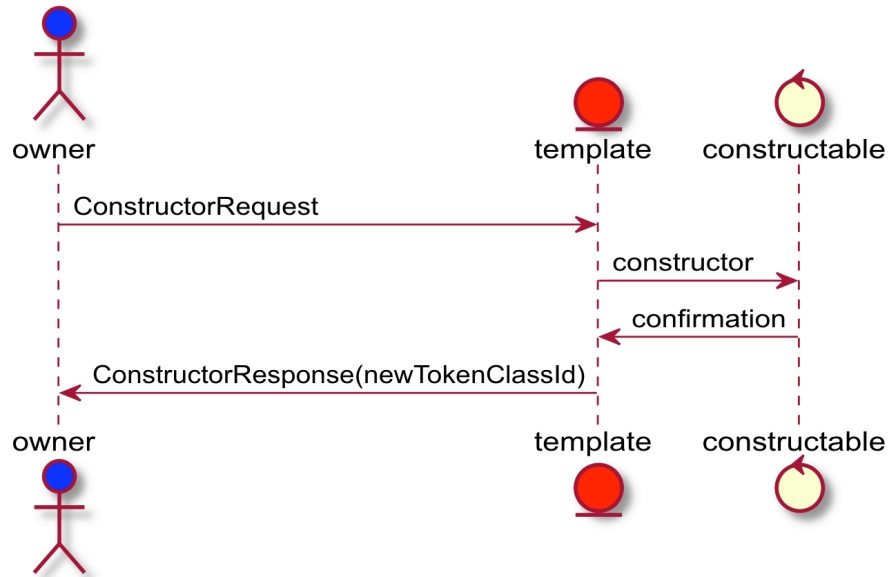


©2019 Token Taxonomy Initiative Inc. (“TTI”). All Rights Reserved.

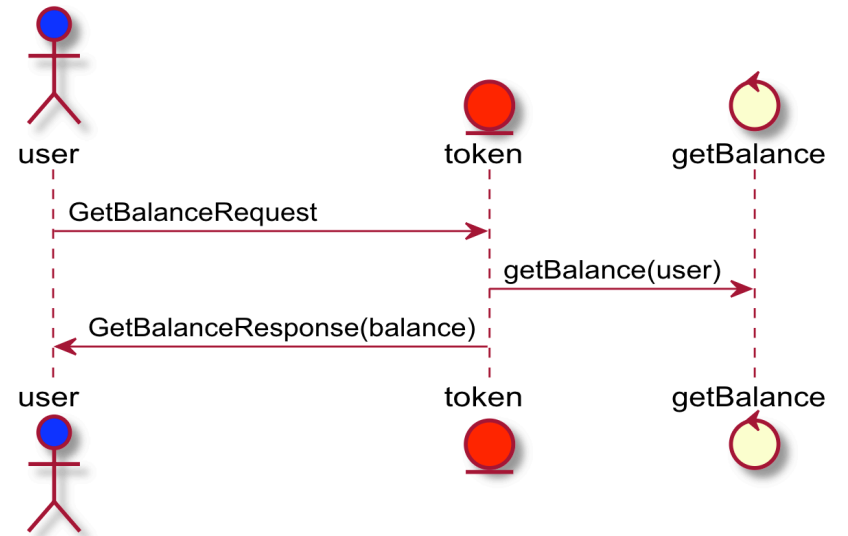
Taxonomy Object Model

- TTF uses GitHub as the backend for storing artifacts.
- Developers can consume the GitHub natively and create artifacts manually.
- The Taxonomy Object Model provides a programmable middle extension for navigating the taxonomy and create/update/delete artifacts.
- This allows for any application to create a presentation, Web, Mobile or Office experience using the model for general and business users.
- Because it stores the data as formatted text, JSON, Proto3 and MD, git hub tracks changes and provides the collaboration tools to facilitate the workshop process.

Create Token



Token Class interaction



Interaction Models

Behavior	Control	Properties
Constructable	Constructor	Name, Symbol, Decimals = 0
Mintable	MintRequest	Quantity = 100
Base	GetBalanceRequest	
SKU Property-Set	GetSKURequest	

