Iken-lang change khagekog

# Changelog

## v1.1.2 - 2024-09-13

### Added

- N/A

### Changed

- \*\*aiken-lang\*\*: Fix issues with static recursive optimization. See [#1009](https://github.com/aiken-lang/aiken/issues/1009) @Microproofs

- \*\*aiken-lang\*\*: Aiken IR now interns variables while building up to ensure uniqueness for local vars. @Microproofs

- \*\*aiken-lang\*\*: Fix reification of `Data` (failing to reify) & `PRNG` (missing variants' arguments). @KtorZ

- \*\*aiken-lang\*\*: Adjust reification of `String` to be shown as plain UTF-8 text strings (instead of hex-encoded byte array). @KtorZ

- \*\*aiken-lang\*\*: Fix formatting of long if-condition over multiline. @KtorZ & @Microproofs

- \*\*aiken-lang\*\*: Fix formatting of standalone logical binary chains (`and` & `or`) in functions. @KtorZ

### Removed

- N/A

## v1.1.1 - 2024-09-10

### Added

- N/A

### Changed

- \*\*aiken-lang\*\*: Fix validator's else handler generation. See [#1015](https://github.com/aiken-lang/aiken/issues/1015) @KtorZ

- \*\*aiken-lang\*\*: Fix underflow in error message reported by the validator arity. See [#1013](https://github.com/aiken-lang/aiken/issues/1013) @KtorZ

- \*\*aiken-lang\*\*: Fix list-pattern needlessly formatting over multiple lines. @KtorZ

- \*\*aiken-lang\*\*: Fix formatter on long alternative patterns spanning over multiple lines. @KtorZ

- \*\*aiken-lang\*\*: Fix needed parentheses under trace-if-false operator for todo, fail, unop & pipelines; removed when formatting. @KtorZ

- \*\*aiken-lang\*\*: Fix formatter removing curly braces around multi-line constants. It's fine to not have curly braces, but it's the Aiken signature after all. @KtorZ

- \*\*aiken-lang\*\*: Improve LSP suggestion for module imports. @Riley-Kilgore

### Removed

- N/A

## v1.1.0 - 2024-09-03

### Added

- \*\*aiken-lang\*\*: also authorize (complete) patterns in function arguments list instead of only variable names. @KtorZ

- \*\*aiken-lang\*\*: new syntax for soft casting otherwise known as `if/is`. See [#959](https://github.com/aiken-lang/aiken/pull/959) or [Control Flow - soft casting](https://aiken-lang.org/language-tour/control-flow#soft-casting-with-ifis) for more details. @rvcas

- \*\*aiken-lang\*\*: optimization: pre-evaluate constant arguments to lambdas when safe to do so. @MicroProofs

- \*\*aiken-lang\*\*: infer type when immediately possible during a patterned type-cast. See [#969](https://github.com/aiken-lang/aiken/pull/979). @KtorZ

- \*\*aiken-lang\*\*: add support for `mk\_cons` and `mk\_pair\_data` builtins. See [#964](https://github.com/aiken-lang/aiken/issues/964). @KtorZ

- \*\*aiken-lang\*\*: pattern-matching on bytearrays is now available. See [#989](https://github.com/aiken-lang/aiken/issues/989). @KtorZ

- \*\*aiken-project\*\*: conditional configuration and environment. See [#937](https://github.com/aiken-lang/aiken/issues/937). @KtorZ

- \*\*aiken-project\*\*: warning on compiler version mismatch. See [de870e2](https://github.com/aiken-lang/aiken/commit/de870e2529eb2336957e228cd30d4850ec2619a2). @rvcas

- \*\*aiken-project\*\*: source links to generated documentation for types, constants and functions. @KtorZ

- \*\*aiken-project\*\*: comments containing Markdown section headings (`#`, `##`, `###` etc.) will now be preserved and rendered in generated documentation. @KtorZ

- \*\*aiken-project\*\*: modules starting with `@hidden` in their docs will be skipped from docs generation. @KtorZ

- \*\*aiken-project\*\*: preserve type-aliases as titles in blueprint generated schemas. @KtorZ

- \*\*uplc\*\*: support evaluation of Plutus V3 transactions, including new purposes introduced in Conway. @KtorZ

### Changed

- \*\*aiken-lang\*\*: zero-arg functions are \*\*no longer\*\* evaluated at compile-time. However, constants can now hold \_any\_ expression and are fully evaluated at compile-time. Use `const` whenever a zero-arg function was used, unless you do want to defer execution. @KtorZ @MicroProofs.

- \*\*aiken-lang\*\*: fix zero-arg builtins `mk\_nil\_data` and `mk\_nil\_pair\_data` invokation. @KtorZ

- \*\*aiken-lang\*\*: rename some builtins. @KtorZ

| old name | new name |

| --- | --- |

| `mk\_nil\_data` | `new\_list` |

| `mk\_pair\_data` | `new\_pair` |

| `mk\_nil\_pair\_data` | `new\_pairs` |

- \*\*aiken-lang\*\*: duplicate import lines are now automatically merged instead of raising a warning. However, imports can no longer appear anywhere in the file and must come as the first definitions. @KtorZ

- \*\*aiken-lang\*\*: remove warning on discarded expect, allowing to keep 'side-effects' when necessary. See [#967](https://github.com/aiken-lang/aiken/pull/967). @KtorZ

- \*\*aiken-lang\*\*: allow expect as last (or only) expression in function body, when clauses and if branches. Such expressions unify with `Void`. See [#1000](https://github.com/aiken-lang/aiken/pull/1000). @KtorZ

- \*\*aiken-lang\*\*: allow tests to return `Void`. Tests that return `Void` are treated the same as tests that return `True`. See [#1000](https://github.com/aiken-lang/aiken/pull/1000). @KtorZ

- \*\*aiken-lang\*\*: rework traces to be (1) variadic, (2) generic in its arguments and (3) structured. @KtorZ

In more details:

1. Enables the `trace` keyword to take one, two or any argument really separated by comma after the first. For example:

```ak

trace @"a classic trace"

// ..

trace @"condition\_1": @"foo"

// ...

trace @"condition\_2": @"foo", @"bar"

```

2. Enables the `trace` keyword to not only take strings as arguments; but any

data-type that is serialisable (i.e. that can be cast to Data). It is fundamentally identical to calling the [`cbor.diagnostic`](https://aiken-lang.github.io/stdlib/aiken/cbor.html#diagnostic) function from the standard lib; except that this is done and glued with the rest of the trace automatically.

```ak

trace @"condition\_1": [1, 2, 3]

// ...

let my\_var = Some("foo")

trace my\_var

```

3. Changes the behavior of the `--trace-level compact` mode to now:

- remove trace-if-false (`?` operator) traces entirely in this mode;

- only keep the label (first trace argument) and error when it isn't a string.

See also [#978](https://github.com/aiken-lang/aiken/pull/978).

- \*\*aiken-lang\*\*: rework formatter behaviour on long-lines, especially in the presence of binary operators. @KtorZ

- \*\*aiken-lang\*\*: provide better errors for unknown types used in cyclic type definitions. @KtorZ

- \*\*aiken-project\*\*: fix blueprint's apply truncating last character of outputs. See [#987](https://github.com/aiken-lang/aiken/issues/987). @KtorZ

- \*\*aiken-project\*\*: provide better error (include input ref) when inputs are missing during transaction evaluation. See [#974](https://github.com/aiken-lang/aiken/issues/974). @KtorZ

- \*\*aiken-project\*\*: module inhabitants are no longer alphabetically sorted when generating documentation. Instead, the order in which they are defined in the module is used. @KtorZ

- \*\*aiken-project\*\*: the sidebar links to modules within a package is now fully hierarchical and (hopefully) better-looking. @KtorZ

### Removed

- \*\*aiken-lang\*\*: clause guards are no longer part of the language. See [#886](https://github.com/aiken-lang/aiken/issues/886). @KtorZ.

## v1.0.29-alpha - 2024-06-06

### Added

- \*\*aiken-lang\*\*: new LSP quickfix for 'use let' warning. @KtorZ

### Changed

- \*\*aiken-lang\*\*: the keyword `fail` on property-based test semantic has changed and now consider a test to succeed only if \*\*every\*\* execution of the test failed (instead of just one). The previous behavior can be recovered by adding the keyword `once` after `fail`. @KtorZ

### Fixed

- \*\*aiken-lang\*\*: fixed the number of 'after x tests' number reported on property test failure, which was off by one. @KtorZ

- \*\*aiken-lang\*\*: fixed parsing of single hex digits. @KtorZ

## v1.0.28-alpha - 2024-05-23

### Added

- \*\*aiken\*\*: install shell completions automatically. @freexploit

- \*\*aiken\*\*: added export command that exports regular function definitons. @rvcas

- \*\*aiken-project\*\*: compiler version field to `aiken.toml` @rvcas

- \*\*aiken-project\*\*: plutus version field to `aiken.toml` @rvcas

- \*\*aiken-lsp\*\*: hover and goto definition support on list tail. @rvcas

- \*\*aiken-lsp\*\*: hover on prop test via expression. @rvcas

- \*\*aiken-lang\*\*: new builtin types in the prelude `Pair` and `Pairs`. @KtorZ @Microproofs

- \*\*aiken-lang\*\*: Codegen now generates uplc version 1.1.0 scripts when running build with plutus v3.

### Fixed

- \*\*aiken-lang\*\*: formatter should not erase `pub` on validators. @rvcas

- \*\*aiken-lang\*\*: error on using tuple index when a tuple is returned by a generic function. @rvcas

- \*\*aiken-lang\*\*: backpassing with expect gives a warning on pattern matches. @rvcas

- \*\*aiken-lang\*\*: fix a regression in the Type-checker introduced in v1.0.25-alpha regarding types comparison. See #917. @KtorZ

- \*\*aiken-lang\*\*: fix incongruous generics after type-checking which caused [] to be treated as a list in cases where it needed to be an empty map primitive. See #922. @KtorZ

- \*\*aiken-lang\*\*: fix for generic constrs being used as functions causing type mismatch errors. @Microproofs

- \*\*aiken-lang\*\*: fix for error occuring when a field holds Data that is not a constr type when compiler traces are on. @Microproofs

- \*\*aiken-lang\*\*: fix for curry optimization involving 2 constants #945. @MicroProofs

- \*\*aiken-lang\*\*: fix compiler wrongly requiring MillerLoopResult to be 'serialisable' when manipulated as a top-level value. See #921. @KtorZ

- \*\*aiken-lang\*\*: fix type-checker oversight regarding serialisation of generics. See #939. @KtorZ

- \*\*aiken-lang\*\*: fix type-checker not raising error when comparing non-serialisable types. See #940. @KtorZ

- \*\*aiken-project\*\*: show a warning when ignoring modules in lib/validator because they have an invalid name. See #916. @KtorZ

### Changed

> [!WARNING]

>

> \*\*BREAKING-CHANGE\*\*

>

> 2-tuples `(a, b)` are now treated the same as 3+ tuples -- which directly impacts the way that Aiken now deserialise those, especially when nested inside a `List`.

>

> To deserialize into a list of 2-tuple (`List<(a, b)>`), one is now expected to provide a CBOR array of arrays (of 2 elements). Previously, this would require to provide a CBOR map! The downside of the latter is that CBOR serialization libraries do not necessarily preserve the order of keys in a map which could cause issues down the line, in particular with Aiken's dictionnaries.

>

> To recover the old behavior when desired, Aiken introduces a new type `Pair<a, b>` to the language. So any existing program can be migrated by switching any occurences of `(a, b)` to `Pair<a, b>`.

>

> However, it is often preferable to use 2-tuples where possible. The main place you will see usage of `Pair` is in the script context because its form is imposed by the ledger.

- \*\*aiken-lang\*\*: altered internal representation of 2-tuples to distinguish them from pairs. @KtorZ @Microproofs

- \*\*aiken-lang\*\*: some more code gen cleanup. @Microproofs

- \*\*aiken-lang\*\*: new optimization for wrapped builtins found in the stdlib. @Microproofs

- \*\*aiken-project\*\*: slightly restyle warnings to be less noisy. @KtorZ

## v1.0.26-alpha - 2024-03-25

### Fixed

- \*\*aiken-lang\*\*: allow casting of types to Data in function pipiing. @KtorZ

## v1.0.25-alpha - 2024-03-22

### Added

- \*\*aiken-lang\*\*: Data now has a generic argument that can be used to specify the blueprint type. @KtorZ

- \*\*aiken-lang\*\*: New types `PRNG` and `Fuzzer` in the prelude. @KtorZ

- \*\*aiken-lang\*\*: Test definitions now accept an (optional) argument alongside a new keyword `via` to specify fuzzers. @KtorZ

- \*\*aiken-project\*\*: Property-based testing framework with integrated shrinking and case labelling. @KtorZ

- \*\*aiken-project\*\*: Unit tests now show assertion operands as Aiken expression instead of raw UPLC . @KtorZ

- \*\*aiken\*\*: The `check` command now accept an extra arg `--seed` to provide an initial seed for the pseudo-random generator of properties. @KtorZ

- \*\*uplc\*\*: add `integerToByteString` and `byteStringToInteger` builtins. @rvcas @Microproofs

- \*\*aiken-lang\*\*: add `integer\_to\_byte\_string` and `byte\_string\_to\_integer` `aiken/builtins`. @rvcas

- \*\*uplc\*\*: more conformance tests for `integerToByteString` and `byteStringToInteger` along with new ones. @rvcas

- \*\*aikup\*\*: error message when version is not found. @rvcas

- \*\*aiken\*\*: support outputting mainnet addresses for validators. @rvcas

- \*\*aiken-lang\*\*: added serde to CheckedModule to encode modules as cbor. @rvcas

- \*\*aiken-lang\*\*: Strings can contain a nul byte using the escape sequence `\0`. @KtorZ

- \*\*aiken\*\*: The `check` command now accept an extra (optional) option `--max-success` to control the number of property-test iterations to perform. @KtorZ

- \*\*aiken\*\*: The `docs` command now accept an optional flag `--include-dependencies` to include all dependencies in the generated documentation. @KtorZ

- \*\*aiken-lang\*\*: Implement [function backpassing](https://www.roc-lang.org/tutorial#backpassing) as a syntactic sugar. @KtorZ

- \*\*aiken-lang\*\*: Extend backpassing to support multiple patterns/arguments. @rvcas

### Fixed

- \*\*aiken-lang\*\*: Boolean operators (`||` and `&&`) were (somewhat) left-associative. This is now fixed and changed to right-associativity. @KtorZ

- \*\*uplc\*\*: `serialise\_data` builtin wrongly encoding some larger ints as tagged CBOR bigints, instead of plain integers over 9 bytes. @KtorZ

- \*\*aiken-project\*\*: Unit tests reports are now inline with the test with less noise. @KtorZ

- \*\*aiken-lang\*\*: Data deserialization for primitive types (pairs, bools, void) now do full checks on the Data structure. @Microproofs

- \*\*aiken-lang\*\*: The identity reducer optimization was not removing the identity function before. That is fixed now.@Microproofs

- \*\*aiken-lang\*\*: Inner opaque types can now be properly destructured by expect and when patterns. @Microproofs

- \*\*aiken-lang\*\*: A codegen specific name-unique interner is now used to preserve lambda scoping. @Microproofs

- \*\*aiken-lang\*\*: if there is only one clause we want to present a warning that suggests that a `let` binding should be used instead but only if it's an exhaustive pattern. @rvcas

- \*\*aiken-lang\*\*: support nested void matching @rvcas

- \*\*uplc\*\*: fix constr identity (`constr-3.uplc`) conformance test. @rvcas

- \*\*aiken-lang\*\*: disallow `MLResult` in a type definition. @rvcas

- \*\*aiken-lang\*\*: reversed deserialization of bls types out of data types. @rvcas

- \*\*aiken-lang\*\*: validator args unexpectedly unbound causing code gen crashes. @rvcas

- \*\*aiken-lang\*\*: allow implicitly discarded values when right-hand side unified with `Void`. @KtorZ

- \*\*aiken-lang\*\*: allow zero arg mutually recursive functions. @Microproofs

- \*\*aiken-lang\*\*: function aliases now resolved to the module and function name in codegen. @Microproofs

- \*\*aiken-lang\*\*: fix indentation of pipelines to remain a multiple of the base indent increment. @KtorZ

- \*\*aiken-lang\*\*: forbid presence of non-serialisable data-types in compound structures like List and Tuple. @KtorZ

- \*\*aiken-lang\*\*: fix 'given' arity reported by 'incorrect arity' error message. @rvcas

### Changed

- \*\*aiken-lang\*\*: Discards will now also type check the validator arguments instead of completely ignoring them. @Microproofs

- \*\*aiken-lang\*\*: Further improvements to tracing when using expect casting from Data. @Microproofs

- \*\*aiken-lang\*\*: The set of curriable builtins with arguments that occur 3 or more times are now hoisted in scope with the arguments curried. @Microproofs

- \*\*aiken-lang\*\*: Improved the way the lambda inliner works to prevent unnecessary inlining into functions. @Microproofs

- \*\*aiken-lang\*\*: Simplifications to the AirTree type in codegen. @Microproofs

- \*\*aiken-lang\*\*: CONSTR\_FIELD\_EXPOSER and CONSTR\_INDEX\_EXPOSER now inline the builtins instead. @Microproofs

- \*\*aiken-lang\*\*: SubtractInteger with a constant as the second arg is now flipped to addInteger with a negated constant. @Microproofs

- \*\*aiken-lang\*\*: Validator arguments are now checked per arg instead of after all args are applied. @Microproofs

- \*\*aiken-project\*\*: remove test definitions from dependency modules. @rvcas

- \*\*aiken-project\*\*: ignore warnings from dependency modules. @rvcas

- \*\*aiken-project\*\*: parse sources in parallel, this resulted in a nice speedup. @rvcas

- \*\*aiken-lang\*\*: You can no longer use expect on opaque types in various situations. @rvcas & @KtorZ

## v1.0.24-alpha - 2024-01-31

### Added

- \*\*aiken\*\*: New aliases for `check` (aiken c) and `build` (aiken b) commands. @Kuly14

### Fixed

- \*\*aiken-lang\*\*: Fixed an issue with expects on lists that used discards. This fixes

the validator issues being seen for previously succeeding validators on 1.0.21-alpha. @MicroProofs

- \*\*aiken-lang\*\*: Out of Span issue is now solved. This also fixes incorrectly selected

traces from the wrong module, which in some cases lead to the out of span issue. @MicroProofs

- \*\*aiken-lang\*\*: Calling head\_list on a list of pairs no longer throws a type error. @MicroProofs

## v1.0.23-alpha - 2024-01-24

### Fixed

- \*\*aiken-lang\*\*: Now using correct discard match for constructor field access.

- \*\*aiken-lang\*\*: The list\_access\_to\_uplc now always returns a lambda wrapped

term even with only discards. This fixes an apply error being seen by a few

people.

## v1.0.22-alpha - 2024-01-24

### Added

- \*\*aiken\*\*: New `--trace-level` option for the `check` and `build` commands to

allow chosing the verbosity level of traces amongst three levels: silent,

compact & verbose. @MicroProofs @KtorZ

- \*\*aiken\*\*: New `--filter-traces` option for the `check` and `build` commands

to enable restricting traces with more granularity between user-defined

- \*\*aiken-lang\*\*: Most builtin errors are now caught and instead catched trace

errors are thrown. The exception is BLS primitives.

### Fixed

- \*\*aiken-lang\*\*: Fix flat encoding and decoding of large integer values. @KtorZ

- \*\*aiken-lang\*\*: Traces should not have following expressions formatted into a

block. @rvcas

- \*\*aiken-lang\*\*: Sequences should not be erased if the sole expression is an

assignment. @rvcas

- \*\*aiken-lang\*\*: Should not be able to assign an assignment to an assignment.

@rvcas

- \*\*aiken-lang\*\*: Should not be able to have an assignment in a logical op

chain. @rvcas

- \*\*aiken\*\*: Ensures that test expected to fail that return `False` are

considered to pass & improve error reporting when they fail. @KtorZ

- \*\*aiken-lang\*\*: Fix generic edge case involving tuples.

- \*\*aiken\*\*: `aiken new` now uses the current version for the github action.

- \*\*aiken-lang\*\*: Using the head\_list builtin on assoc lists now works.

### Removed

- \*\*aiken\*\*: The options `--keep-traces` (on the `build` command) and

`--no-traces` (on the `check` command) have been removed; superseded by the

new options. @MicroProofs @KtorZ

> [!TIP]

>

> - If you've been using `aiken check --no-traces`, you can recover the old

> behavior by doing `aiken check --trace-level silent`.

> - If you've been using `aiken build --keep-traces`, you can recover the old

> behavior by doing `aiken build --trace-level verbose`.

## v1.0.21-alpha - 2023-12-04

### Added

- \*\*aiken\*\*: `--watch` flag on the `build`, `check` and `docs` commands to

automatically watch and re-execute the command on file changes.

@Quantumplation & @KtorZ

- acceptance tests 28-30 @MicroProofs

- \*\*aiken-lang\*\*: expose BLS builtins and types @MicroProofs & @rvcas

- \*\*aiken-lsp\*\*: implement hover info for tuples, lists, and contructor pattern

elements @rvcas

- \*\*aiken-lsp\*\*: implement hover on when clause patterns @rvcas

- \*\*aiken-lsp\*\*: hover support for the optional multi validator fn @rvcas

- \*\*aiken-lsp\*\*: implement quickfix for "utf8 byte array is valid hex string"

warning @rvcas

- \*\*uplc\*\*: add all BLS builtins and types @MicroProofs & @rvcas

- \*\*uplc\*\*: add plutus conformance tests from

[here](https://github.com/input-output-hk/plutus/tree/master/plutus-conformance/test-cases/uplc/evaluation).

@MicroProofs & @rvcas

- \*\*uplc\*\*: case and constr cost models @MicroProofs

### Changed

- \*\*aiken\*\*: update to pallas v0.20.0 @rvcas

- \*\*aiken-project\*\*: switch blueprint validator tests now uses insta

@MicroProofs

- \*\*aiken-project\*\*: update to pallas v0.20.0 @rvcas

- \*\*aiken-lang\*\*: use a better algorithm for inlining single occurrences

@MicroProofs

- \*\*uplc\*\*: update to pallas v0.20.0 @rvcas

- \*\*uplc\*\*: remove `flat-rs` crate and use it through pallas\_codec instead

### Fixed

- \*\*aiken-lang\*\*: formatting unable to have newline after expect bool shortcut

@rvcas

- \*\*aiken-lang\*\*: formatter incorrectly erasing blocks in certain situations

@rvcas

- \*\*aiken-lang\*\*: use a distinct warning for discarded let assignments to avoid

confusion @rvcas

- \*\*aiken-lang\*\*: allow spread operator on positional constructor arguments

@rvcas

- \*\*aiken-lsp\*\*: and/or chains hovering on nested elements not working @rvcas

- \*\*uplc\*\*: delay typemismatch errors in the machine runtime (caught by

conformance tests) @rvcas

## v1.0.20-alpha - 2023-10-25

### Added

- \*\*aiken-project\*\*: The `plutus.json` blueprint now contains a `compiler.name`

and `compiler.version` fields.

- \*\*aiken-prokect\*\*: Added compiler and system information to panic error

report.

- \*\*aiken-lsp\*\*: Added quickfix suggestions for unknown variables, modules and

constructors.

### Changed

- \*\*aiken-lang\*\*: Added validator specific error when validator returns false

- \*\*aiken-lang\*\*: Running a multi-validator with traces on will let you know

which validator function is running

- \*\*aiken-lang\*\*: Multi-validator with traces will give a better error when the

'spend' redeemer is not properly wrapped with a constr

- \*\*aiken-lang\*\*: Code gen traces are referenced instead of being inlined

leading to smaller validators

### Fixed

- \*\*aiken-lang\*\*: improved error messages for `a |> b(x)`.

- \*\*uplc\*\*: Fixed cost model issue when using div, quot, rem, mod.

- \*\*aiken-lsp\*\*: Improved hovering suggestions and type annotations.

## v1.0.19-alpha - 2023-09-29

### Fixed

- \*\*aiken-lang\*\*: Fixed small typo in `hoist\_dependent\_functions` lead to

internal cyclic calls not being recognized.

## v1.0.18-alpha - 2023-09-29

### Added

- \*\*aiken-lang\*\*: Code gen now allows for mutual recursion

### Fixed

- \*\*aiken-lang\*\*: fixed stack overflow with unbound typing being passed into a

function with inferred types

- \*\*aiken-lang\*\*: expect on tuples greater than 2 checks the length to ensure

tuple lists are the same length as the type.

### Changed

- \*\*aiken-lang\*\*: (Code Gen): Rename some of the types to use aliases

- \*\*aiken-lang\*\*: (Code Gen): Remove the use of Air::RecordAccess and TupleIndex

and replace them with functions that directly get the item at the specified

index. Also improves performance.

- \*\*uplc\*\*: Added more cases to the inline optimization function to allow the

removal of further unnecessary lambda bindings.

## v1.0.17-alpha - 2023-09-20

### Added

- \*\*aiken\*\*: add ability to force warnings to cause a failing exit code on

check, build, and docs

- \*\*aiken-lang\*\*: automatically resolve and fetch latest revision of a package

on build when a branch is specified as version

- \*\*uplc\*\*: Add Case and Constr Terms; This includes their flat serialization

and evaluation

### Fixed

- \*\*uplc\*\*: trim whitespace when loading files with hex strings to avoid

confusing errors #720

- \*\*uplc\*\*: uplc `Constant::Data` formatting

- \*\*aiken-lang\*\*: code gen fixes including nested constr when matches and expect

on None

- \*\*aiken-lang\*\*: empty records properly parse as record sugar

- \*\*aiken-lang\*\*: escape sequences are now properly preserved after formatting

- \*\*aiken-lang\*\*: fixed parser ambiguity when using record constructor in if

conditions followed by single-line var expressions #735

- \*\*aiken-project\*\*: when a module name has a hyphen we should behave like rust

and force an underscore

## v1.0.16-alpha - 2023-08-24

### Removed

- \*\*aiken\*\*: `aiken new` no longer accept an `--empty` flag. Projects are

generated empty by default.

## v1.0.15-alpha - 2023-08-19

### Added

- \*\*aiken\*\*: Parameters for `blueprint apply` can now be built interactively.

### Changed

- \*\*aiken-lang\*\*: Opaque types are now properly handled in code gen (i.e. in

code gen functions, in datums/redeemers, in from data casts).

- \*\*aiken\*\*: `blueprint apply` now expects only one OPTIONAL argument. When not

provided, the parameter will be prompted interactively.

- \*\*aiken-lang\*\*: New tests for code gen around opaque types.

- \*\*aiken-lang\*\*: `traverse\_with\_tree` now has a boolean parameter to determine

when `with` is called.

### Removed

- \*\*aiken\*\*: `blueprint apply` no longer accept a target directory. The command

has to be executed within the same directory as the `aiken.toml`.

## v1.0.14-alpha - 2023-08-16

### Added

- \*\*aiken\*\*: `new` command now fetches latest stdlib version

- \*\*aiken-lang\*\*: new `and` and `or` chaining

```

and {

1 == 1,

or {

2 == 2,

3 != 2,

True,

},

}

```

### Changed

- \*\*aiken-lang\*\*: significantly improved pattern matching exhuastiveness

checking

- \*\*aiken-lang\*\*: Tx Simulate now returns a list of execution budgets for each

redeemer instead of calculating the total units required to run all the

scripts.

- \*\*aiken-lang\*\*: Now code gen uses a tree abstraction to build the Aiken

Intermediary Representation. This now fixes quite a number of minor issues

while making the code more maintainable. This is a large leap towards a stable

version and future updates will be on further simplifying and stabilizing the

tree abstraction.

- \*\*aiken-lang\*\*: Zero argument anonymous functions now are implemted as a

delayed function body and calling them simply does force

- \*\*aiken-lang\*\*: Matching on int in expect and when cases is now implemented

- \*\*aiken-lang\*\*: Using assign in nested pattern matches is now implemented

- \*\*aiken-lang\*\*: Using List<Data> as a validator params only checks the type is

a list and does not attempt to check each item

- \*\*aiken-lang\*\*: Recursion optimization to prevent static parameters from being

passed through every recursion

- \*\*aiken-lang\*\*: aliased import of single type throws compiler error

- \*\*aiken-lsp\*\*: fix diagnostics and formatting on windows vscode

- \*\*aiken\*\*: decode should always print to textual

- \*\*uplc\*\*: pair type formatting

## v1.0.13-alpha - 2023-07-15

### Added

- \*\*aiken-lang\*\*: `expect foo == bar` desugars into `expect True = foo == bar`

### Fixed

- \*\*aiken-lang\*\*: fail, todo, and trace had issues with sequences and

expressions

## v1.0.12-alpha - 2023-07-14

### Added

- \*\*aiken\*\*: added a `blueprint policy` command to compute the policy ID of a

minting script

- \*\*uplc\*\*: parsing and pretty printing for PlutusData

### Fixed

- \*\*aiken-lang\*\*: Prevent mutual recursion caused by conflicting function names

for generic expect type

- \*\*aiken-lang\*\*: UPLC evaluation of large integers literals (> max u64)

- \*\*aiken-lang\*\*: Parsing of error / todo keywords in when cases

- \*\*aiken-lang\*\*: Parsing of negative integer patterns and constants

- \*\*aiken-lang\*\*: automatically infer unused validator args as `Data`

- \*\*aiken-lang\*\*: test crashing when referencing validators

- \*\*aiken\*\*: mem and cpu values were not showing in white terminals, switched to

cyan

### Changed

- \*\*uplc\*\*: make list type and pair type parsing match the plutus core spec

- \*\*aiken-lang\*\*: rename `error` to `fail`

- \*\*aiken-lang\*\*: new failing test syntax `test name() fail {`

### Removed

## v1.0.11-alpha - 2023-06-23

### Added

- \*\*aiken\*\*: enhance `aiken new`

- \*\*aiken-lang\*\*: Binary operator are now treated like first-class citizen in

expressions. In particular, they can be used as function arguments directly:

```

compare\_with(a, >=, b) == compare\_with(a, fn(l, r) { l >= r }, b)

```

- \*\*aiken-lang\*\*: Make traces produced by expect dependent on the value of the

tracing flag.

### Fixed

- \*\*aiken-lang\*\*: Explain discards and expect a bit better in the unused var

warning

- \*\*aiken-lang\*\*: Fix expect \\_ = ... not including the cast from data logic if

the type is data and right hand has a type annotation

- \*\*aiken-lang\*\*: Fix for the final clause of a when expecting another clause

afterwards in nested list cases.

- \*\*aiken-lang\*\*: Fix for all elements were being destructured in tuple clauses

even if not used.

- \*\*aiken-lang\*\*: Fix for tuple clause not consuming the next case causing

incomplete contracts. Now tuple clause will always consume the next case

unless it is the final clause

- \*\*aiken-lang\*\*: Fix for builtins using the incorrect data to type conversion

when used as a function param.

## v1.0.10-alpha - 2023-06-13

### Added

- \*\*aiken\*\*: added panic hook to present a link to make a bug report

### Fixed

- \*\*aiken-lang\*\*: fmt for `module.Constr { field: value }`

- \*\*aiken-lang\*\*: Issue where using var pattern in a when was passing the constr

index instead of the constr

- \*\*aiken-lang\*\*: Issue where expecting on a list had unexpected behaviors based

on list length.

- \*\*aiken-lang\*\*: Issue where expecting on a list from data was using the wrong

expect cast for the list tail.

### Removed

N/A

## v1.0.8-alpha - 2023-06-08

### Added

- \*\*aiken-lang\*\*: numbers can now be written as hexadecimal digits (e.g. `0x42`)

- \*\*aiken-lang\*\*: numbers can now be written using numeric underscores as

separator (e.g. `1\_000\_000`)

### Fixed

- \*\*aiken-lang\*\*: fixed operator precedences, in particular the pipe operator

(`|>`) which is now of the lowest precedence.

- \*\*aiken-project\*\*: need to convert to Program<Name> before dumping to uplc

- \*\*aiken-lang\*\*: fmt crashing when comment at end of file with no newline

- \*\*aiken-lang\*\*: Fixed error when using nested boolean checks in when

conditions

- \*\*aiken-lang\*\*: Had the wrong conversion for constant maps to plutus data.

Fixed to check for right conversion

- \*\*aiken-lang\*\*: Zero arg functions were grabbing extra dependencies they

didn't need to

- \*\*aiken-lang\*\*: Rearrange list clauses and fill in gaps now handles nested

patterns in a uniform way

- \*\*aiken-lang\*\*: Fixed discards in records that were being sorted incorrectly

leading to type issues

### Removed

N/A

## v1.0.7-alpha - 2023-06-02

### Added

- \*\*aiken-lang\*\*: Add a way to express that tests can return an error

### Fixed

- \*\*uplc\*\*: Fix pair formatting

- \*\*aiken-lang\*\*: forced new line in formatter for assignments

- \*\*aiken-lang\*\*: Incorrect parsing of generic type annotation prefixed with

module

- \*\*aiken-lang\*\*: Incorrect handling of comments at end of a file when newline

not present

- \*\*aiken-lang\*\*: Record update in code gen is now flexible enough to support

fields being passed in any order.

- \*\*aiken-lang\*\*: Record update now produces better uplc code then creating a

record by the normal instantiation.

- \*\*aiken-lang\*\*: Issue with Constructors being passed as functions to other

function arguments was fixed.

- \*\*aiken-lang\*\*: show module name when type mismatch names are the same

- \*\*aiken\*\*: make `uplc flat` and `uplc unflat` subcommands more consistent and

flexible

## v1.0.6-alpha - 2023-05-17

### Fixed

- \*\*aiken-lang\*\*: Fix for cases where identity function is used as a param to a

function or assigned to a var.

- \*\*aiken-lang\*\*: Fix for free unique caused by code gen function having a

missing dependency.

## v1.0.5-alpha - 2023-05-14

### Added

- \*\*aiken-lang\*\*: Add Record Module support for records with 0 fields

- \*\*aiken-lang\*\*: Added some optimization tests

- \*\*aiken-lang\*\*: Added an Aiken to uplc conversion tests on validators

- \*\*aiken-lang\*\*: Added some uplc builder functions for builtin creation

- \*\*aiken-lang\*\*: Added optimization where identity functions called on an arg

are reduced to just the arg.

### Fixed

- \*\*aiken-lang\*\*: Fix for nested constructors where the type had a single

constructor. The fields exposed were not being added to the Air.

## v1.0.4-alpha - 2023-05-09

### Added

- \*\*aiken-lang\*\*: ChooseUnit builtin uses a more efficient way of handling the

first arg (unit) by just assigning to lambda

### Fixed

- \*\*aiken-lang\*\*: Negative numbers now show up as a constant instead of 0 - that

number

- \*\*aiken-lang\*\*: Expect on constructors without field maps no longer panics

- \*\*aiken-lang\*\*: Expect on constructors with discard as assigned field names

now no longer throws free unique

### Changed

- \*\*aiken-lang\*\*: Refactor how builtins are processed to uplc in code gen

## v1.0.3-alpha - 2023-04-28

### Added

- \*\*aiken-lang\*\*: added optimization to help prevent unnecessary data wraps or

unwraps

- \*\*aiken-lang\*\*: optimization to strip unnecessary lambdas

- \*\*aiken-lang\*\*: implement Clone for ParseError and tipo::Error for the

playground

- \*\*aiken-project\*\*: added end to end tests on conversion from aiken lang to

uplc

### Fixed

- \*\*aiken\*\*: fixed 'new' instructions to properly show project name and folder

- \*\*aiken-lang\*\*: Add name of var to the unused var warning

- \*\*aiken-lang\*\*: fix expect on an empty list

- \*\*aiken-lang\*\*: pattern match on boolean with simple clause bodies

- \*\*aiken-lang\*\*: fix for inline\_direct\_reduce to be applied to pattern match

function instead of argument

- \*\*aiken-lang\*\*: code gen function dependencies won't be hoisted to the top,

instead hoisted at the location depended on.

## v1.0.2-alpha - 2023-04-17

### Fixed

- \*\*aiken-lang\*\*: needed to assert\_no\_assignment in when and if blocks

- \*\*uplc\*\*: need to return deserialization error in some cases for the machine

builtins

## v1.0.1-alpha - 2023-04-16

### Fixed

- \*\*aiken-lang\*\*: incorrect scoping for anonymous functions

- \*\*aiken-lang\*\*: duplicate arguments were allowed in anonymous functions

## v1.0.0-alpha - 2023-04-13

### Added

- \*\*aiken\*\*: new command `blueprint convert`

### Changed

- \*\*aiken-project\*\*: tests filtering with `-m` during check now happens in

`Project::collect\_tests`

- \*\*aiken-project\*\*: fixed generation of blueprints for recursive and mutually

recursive data-types

- \*\*aiken-project\*\*: perform validation of parameters on `blueprint apply`

- \*\*aiken-lang\*\*: block `Data` and `String` from unifying when casting

- \*\*aiken-lang\*\*: remove ability for a type with many variants with matching

field labels and types to support field access

- \*\*aiken-lang\*\*: various uplc code gen fixes

- \*\*aiken-lang\*\*: update todo warning to include type

- \*\*aiken-lang\*\*: `|>` operator can now be formatted as a single (short) line or

forced over multiline in a flexible manner

- \*\*aiken-lang\*\*: the compiler now provides better feedback for type holes (i.e.

`\_`) in type annotations

- \*\*aiken-lang\*\*: assignment and clause guard are now always formatted on a new

line

- \*\*aiken-lang\*\*: unused let-bindings are now fully removed from generated code

and discarded unused let-binding now raise a warning

- \*\*aiken-lang\*\*: support multi-clause patterns (only as a syntactic sugar)

- \*\*aiken-lang\*\*: fix lexer panic when parsing too large (> u32) tuple-indexes

- \*\*uplc\*\*: Greatly improved the Plutus virtual machine performances for script

evaluation

## v0.0.29 - 2023-02-23

### Added

- \*\*aiken-project\*\*: new dep rayon for parallel test execution

- \*\*aiken\*\*: new blueprint command

- \*\*aiken-lang\*\*: new syntax for defining validators

- \*\*aiken\*\*: new address command for deriving addresses out of `plutus.json`

- \*\*aiken-lang\*\*: Add missing Plutus builtins to Aiken's lang.

- \*\*aiken\*\*: fancy nix stuff

- \*\*aiken-lsp\*\*: go to definition

- \*\*aiken-lsp\*\*: docs on hover

- \*\*aiken-lsp\*\*: enable compiler a project

### Changed

- \*\*aiken-lang\*\*: `assert` renamed to `expect`

- \*\*aiken-lang\*\*: new syntax for strings and byte array literals

- \*\*aiken-lang\*\*: lots of code gen improvements

- \*\*aiken-lang\*\*: validator checks now happen during infer instead of in project

- \*\*aiken-lang\*\*: fixed unicode parsing

- \*\*aiken-lang\*\*: update default costs models

- \*\*aiken-lang\*\*: Use variable-length threshold for levenshtein distance

- \*\*aiken-project\*\*: Move module name validation outside of type-checking

- \*\*aiken-project\*\*: Add 'plutusVersion' to blueprints

### Removed

- \*\*aiken-project\*\*: remove assets folder in favor of `plutus.json`

- \*\*aiken-lang\*\*: removed some unused constant related data types

## v0.0.28 - 2023-01-06

### Added

N/A

### Changed

- \*\*uplc\*\*: Reward accounts are now correctly turned into script credentials in

ScriptContext.

- \*\*all\*\*: bump pallas version to `v0.16.0`

### Removed

N/A

## v0.0.27 - 2022-MM-DD

### Added

- \*\*aiken-lang\*\*: integrated unit tests

Aiken now supports writing unit tests directly in source files using the new

`test` keyword. Tests are functions with no arguments that are implicitly

typed to `bool`. For example:

```gleam

test foo () {

1 + 1 == 2

}

```

- \*\*aiken\*\*: new `--skip-tests` flag for the `check` command

### Changed

- \*\*aiken\*\*: `check` now also runs and reports on any `test` found in the

project

- \*\*aiken\*\*: fix Plutus V1 `to\_plutus\_data()` for post-alonzo txout with no

datum hash

### Removed

N/A

## v0.0.26 - 2022-11-23

### Added

- \*\*aiken-lsp\*\*: handle `DidSaveTextDocument` notification

- \*\*aiken-lsp\*\*: convert errors into `lsp\_types::Diagnostic`

- \*\*aiken-lang\*\*: doc comment parsing

- \*\*aiken-lang\*\*: code generation for pattern matching expressions

- \*\*aiken-lang\*\*: extended script context

- \*\*aiken-lang\*\*: added Option to builtins

- \*\*aiken-lang\*\*: properly handle record parsing and sugar in patterns

## v0.0.25 - 2022-11-14

### Added

- \*\*aiken\*\*: new `lsp` command

- \*\*aiken\*\*: new `fmt` command

- \*\*aiken\*\*: `build` command now works and outputs assets

- \*\*aiken\*\*: validate project name on `aiken new`

- \*\*aiken-lang\*\*: formatter for `UntypedExpr`

- \*\*aiken-lang\*\*: uplc code gen

- \*\*aiken-lang\*\*: add `Data` to prelude

- \*\*aiken-lang\*\*: allow `Data` to unify with anything that's not in the prelude

- \*\*aiken-project\*\*: validate if validator function return bool

- \*\*aiken-project\*\*: validate if validator function has minimum number of

arguments

- \*\*aiken-lsp\*\*: new crate that contains the aiken language server

### Changed

- \*\*uplc\*\*: `Converter::get\_index` now takes the full name to provide better

error messages for `Error::FreeUnique`

## v0.0.24 - 2022-11-04

### Changed

- \*\*uplc\*\*: Sorted remaining structured in the ScriptContext (Value, Wdrl, (Ref)

Inputs, Mint, Required signers, Data, Redeemers)

## v0.0.23 - 2022-11-03

### Changed

- \*\*uplc\*\*: sort inputs for script context fixes an issue in lucid

https://github.com/spacebudz/lucid/issues/109

## v0.0.22 - 2022-10-31

### Added

- \*\*aiken\*\*: Fancy errors using [miette](https://github.com/zkat/miette)

- \*\*aiken\*\*: Typechecking

- \*\*aiken\*\*: Inject `aiken/builtin` module with some functions from

`DefaultFunction` in UPLC directly exposed

- \*\*aiken-lang\*\*: add `infer` method to `UntypedModule` which returns a

`TypedModule`

- \*\*uplc\*\*: Expose various Pallas primitives from UPLC to make constructing UPLC

types possible for consumers

### Changed

- \*\*aiken\*\*: Project structure is now a bit different. See

[examples/sample](https://github.com/aiken-lang/aiken/tree/main/examples/sample)

for more

## v0.0.21 - 2022-10-23

### Added

- \*\*flat-rs\*\*: New errors for debugging flat decoding issues

### Changed

- \*\*uplc\*\*: Fixed overflow issue by changing `i64` to `i128` in `BigInt::Int`

instances

- \*\*uplc\*\*: Added `apply\_params\_to\_script` function (applies params to script

and serializes the new script).

## v0.0.20 - 2022-10-17

### Added

- \*\*aiken\*\*: `Project` module which is responsible loading modules and running

the compilation steps

- \*\*aiken\*\*: `UplcCommand::Flat` flip the cbor\_hex if condition so that the

correct logic runs when using the flag

- \*\*uplc\*\*: use i128 for `Constant::Integer`

- \*\*flat-rs\*\*: add support for i128 encode and decode

- \*\*flat-rs\*\*: add i128 zigzag function