On Ink Integration Tests 2

Milestone Report

October, 2023

Milestone 1: Execution and Further Analysis

Timeline

Started on 2023-09-25, ending on 2023-10-27.

Total duration: 4 weeks.

What was defined

Definitions below correspond to the deliverables proposed in our grant application CoinFabrik_On_Ink_Integration_Tests_2.

This analysis is based on the commit c2af39883aab48c71dc09dac5d06583f2e84dc54 of the ink! repository.

1. **Documentation.**

   We will write a comprehensive report that compares the functionalities of integration tests and E2E (End-to-End) tests. This report will focus on the functions to be implemented/corrected in this milestone, corresponding to issues `1-default_accounts()`, `2-set_contract_storage()` and `7-instantiate_contract()`.

   Documentation and test cases will be provided for the 13 functions with remaining analysis. If implementation differences are found in these functions, an estimate for their correction and an implementation idea will also be provided in our report.

   If applicable, we will suggest additional tests outside of the scope of this milestone. Particularly, for functions declared outside of the `env_access.rs` file, but that could be related to integration or e2e testing.

2. **Testing and Testing Guide.** The newly developed functionalities will be documented and tested following existing contribution guidelines. A testing guide will be included.

3. **Article.** We will prepare a summary report and publish it on our blog https://blog.coinfabrik.com/.

4. **Develop**.

   We will develop the missing functionalities or correct implementation differences for functions `1-default_accounts()`, `2-set_contract_storage()` and `7-instantiate_contract()`. All these implementations or modifications will be pushed into the ink! project repository following existing contribution guidelines.

If applicable, we will develop additional tests or make ad-hoc improvements to the ink codebase necessary for the completion of this milestone. Particularly for functions declared outside the env_access.rs file that might be related to integration or end-to-end testing.

5. **Review and Estimate.** We will review the remaining 13 unanalysed functions, which are implemented both for integration and e2e tests. For these functions we will provide documentation, a test case and an implementation estimation if applicable. These correspond to functions issue numbers 12 through 24.

6. **Quality Assurance.** We will adhere to existing contribution guidelines and add necessary tests to integrate the new implemented or corrected functions to the ink! project repository.

What we did

1. **Documentation.** We worked on the comparison report, finishing on time. Check the report included as *Appendix 1*. Check the documentation of the 13 functions with pending analysis in our project repository.

2. **Testing and Testing Guide.** We followed contribution guidelines for the pull requests associated with the implementations of functions: `default_accounts()`, `set_contract_storage()` and `instantiate_contract()`. For these functions, we added test cases in their target directories correspondingly. We include a testing guide on how to execute these tests in the *Reference 1: Implementation Summary* of section *Execution* of the *Appendix 1*, and in the documentation of each corresponding pull request.

3. **Article.** Check our summary report published in our blog.

4. **Develop.** We provide implementations for the three functions specified in this milestone: `default_accounts()`, `set_contract_storage()` and `instantiate_contract()`. On the other hand, as we worked on integrating our test cases into CI/CD, we identified a bug when building e2e tests in a workspace. Concretely, we noticed that the build for e2e tests failed when running cargo-contract inside a workspace package, and if any dependency was inherited from the workspace definition. We performed a pull request to the cargo-contract repository implementing a fix to this issue. Check documentation on these developments in the subsection *Ad-hoc developments* of the section *Execution* of the *Appendix 1.*

5. **Review and Estimate.** We performed a review of the 13 functions with pending analysis and found implementation differences between integration and e2e tests for the functions `balance()` and `weight_to_fee()`. For `balance()` we provide an implementation estimate. For `weigh_to_fee()` we provide documentation of the problem and an associated test case; the issue occurs on the e2e environment and needs to be discussed with the responsible team before moving forward. Check

documentation of these reviews and estimates on *Reference 2: Findings and Estimation* of section *Further Analysis and Estimation* of the *Appendix 1.*

6. **Quality Assurance.** We reviewed and followed the established contribution guidelines for the three pull requests performed: PR1-default_accounts(), PR2-set-account_storage(), PR2-instantiate-contract().

Appendix 1: Report on ink! integration tests, execution and further analysis

Abstract

Integration tests run significantly faster than e2e tests. If a full range of functionalities were provided, it could reduce testing and QA times. Ideally, each test in e2e should have its counterpart in Integration.

We have discovered that integration tests for ink! contracts lack some of the functionalities that are available in e2e testing via the natural functionalities provided by a node.

As part of a previous grant application CoinFabrik_On_Ink_Integration_Tests, we divided the workload into 24 functions exposed for their usage in ink! integration and e2e testing environments, and performed a thorough review of 11 of these functions (the first set). Of the 11 functions in this first set, we deemed the implementation of 2 of them unfeasible, and provided an implementation order and implementation estimates for the remaining 9 functions.

During the four weeks corresponding to the current milestone CoinFabrik_On_Ink_integration_Tests_2, we performed a full review of the 13 functions with pending analysis (the second set) and implemented 3 of the 9 functions.

The functions implemented in this milestone were the functions `1-default_accounts()`, `2-set_contract_storage()` and `7-instantiate_contract()`. The first two functions were selected due to their expected straightforward implementation, while `instantiate_contract()` was selected due to the fact that many of the remaining functions with estimations depended on its prior implementation. For these developments, pull requests have been performed to the relevant ink! repositories following contribution guidelines (PR1-default_accounts(), PR2-set-account_storage(), PR2-instantiate-contract().). We also incorporated our test cases to these repositories, showing that the new implementations implement the missing functionalities or correct the differences observed as

part of our initial research. Furthermore, we performed a contribution to cargo-contract in order to allow the incorporation of e2e tests in continuous integration.

On the other hand, we performed a review of the remaining 13 functions (2nd set), providing documentation and test cases for each function. We found behavior and implementation differences between integration and implementation for functions `weight_to_fee()`, and `balance()`. We provide details on the work to be performed on these functions in *Reference 2: Findings and Estimations* of the *Further Analysis and Estimation* section of *Appendix 1*, adding them to the list of functions to be implemented in our next milestone.

Scope and Initial Status

For the implementation and further analysis carried out in this milestone, we focused on the 24 functions exposed for their usage in integration and e2e tests in the `env_access.rs` file.

Table 1: Exposed Functions

Test Type	Exposed Functions	Reviewed Commit
Integration	https://github.com/paritytech/ink/blob/master/crates/ink/src/env_access.rs	c2af39883aab48c71dc09dac5d06583f2e84dc54
E2E		

In the table below, we enumerate these functions, and mention their initial status before the commencement of this milestone. The Issue Number represents the order in which we initially reviewed these functions.

Table 2: Initial Status

Issue Number	Function	Implemented Integration Tests	Implemented Integration E2E Tests	Initial Status
1	`default_accounts()`	Yes	Yes	Implementation Difference.
2	`set_contract_storage()`	Yes	Yes	Missing limitation on Integration Testing.
3	`invoke_contract_delegate()`	No	Yes	Missing Function Implementation on Integration Testing.
4	`invoke_contract()`	No	Yes	Missing Function

				Implementation on Integration Testing
5	`gas_left()`	No	Yes	Missing Function Implementation on Integration Testing. Unfeasible Implementation.
6	`set_code_hash()`	No	Yes	Missing Function Implementation on Integration Testing.
7	`instantiate_contract()`	No	Yes	Missing Function Implementation on Integration Testing.
8	`caller_is_origin()`	No	Yes	Missing Function Implementation on Integration Testing.
9	`code_hash()`	No	Yes	Missing Function Implementation on Integration Testing.
10	`own_code_hash()`	No	Yes	Missing Function Implementation on Integration Testing.
11	`call_runtime()`	No	Yes	Missing Function Implementation on Integration Testing. Unfeasible Implementation.
12	`caller()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
13	`transferred_value()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
14	`weight_to_fee()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
15	`block_timestamp()`	Yes	Yes	Pending Analysis

				for Corrections in Implementation Differences.
16	`account_id()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
17	`balance()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
18	`block_number()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
19	`minimum_balance()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
20	`terminate_contract()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
21	`transfer()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
22	`hash_bytes()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
23	`hash_encoded()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.
24	`ecdsa_recover()`	Yes	Yes	Pending Analysis for Corrections in Implementation Differences.

Execution

We performed the implementation of the selected functions and performed ad-hoc developments adhering to CI/CD best practices.

Implementation of Selected Functions

We performed the implementation of missing functionalities or corrected implementation differences for functions `1-default_accounts()`, `2-set_contract_storage()` and `7-instantiate_contract()`. In the tables below, we summarize the relevant data associated with their development and provide a short testing guide.

Reference 1: Implementation Summary

Function	`default_accounts()`
Initial Status	Implementation Difference
Issue Description	Default account addresses do not match across integration and e2e tests.
Issue Documentation & Test Case	https://github.com/CoinFabrik/on-ink-integration-tests/tree/main/test-cases/default-accounts
Current Status	Implementation Difference Corrected. Test Cases in Target Repo Passed. Pull Request Performed to Corresponding Repository.
Implementation Notes	Now the integration tests mimic the account setup in e2e tests. We changed the name of the accounts "Django" to "Dave" and "Frank" to "Ferdie". On the other hand, there were two accounts in e2e that did not exist in integration tests, accounts “one” and “two”. We added these accounts to integration tests. Moreover, since e2e tests were drawing these accounts from the library `sp_keyring::sr25519::Keyring`, we made integration tests depend on the same library in order to account for future changes in this lib.
Updated Documentation	We updated the original documentation of this issue in our repository, adding the section Update on Correcting this Issue and informing of the correction. In the pull request to the target directory, we updated the inline documentation. The associated documentation should be updated automatically in these pages:

	• https://paritytech.github.io/ink/ink_env/test/fn.default_accounts.html • https://paritytech.github.io/ink/ink_env/test/struct.DefaultAccounts.html • https://paritytech.github.io/ink/ink_e2e/enum.AccountKeyring.html
Testing Guide	In the directory `integration-tests/default_accounts` of the target directory, we include in our pull request a test case showing that the observed implementation difference has been corrected. Note that this test is different from the original test case in our repository, which showed the implementation difference. In this directory `integration-tests/default_accounts`, run the following command to run both integration and e2e tests: `cargo test --features e2e-tests` These tests are run in the same function in order to compare the results of both environments.
Link to Pull Request	PR1-default_accounts()

Function	`set_contract_storage()`
Initial Status	Missing limitation for storage size on integration testing.
Issue Description	The storage in the integration environment does not have the same limitation as in the e2e environment. In the e2e environment, the storage size is limited by 16380 bytes.
Issue Documentation & Test Case	https://github.com/CoinFabrik/on-ink-integration-tests/tree/main/test-cases/set-contract-storage
Current Status	Missing Limitation Implemented. Test Cases Passed. Pull Request Performed to Corresponding Repository.
Implementation Notes	The function `set_contract_storage()` sets the storage in a contract. There was a missing validation in integration tests that was present in e2e. This validation checked that the `size` of the storage set did not exceed 16380 bytes. We added a validation to the function `set_contract_storage()` in integration tests that checks

	the size of the input value against the same limit set in e2e test: 16380 bytes. In case this limit is exceeded, a panic is raised with the message: "Value too large to be stored in contract storage, maximum size is {} bytes".
Updated Documentation	We updated the original documentation of this issue in our repository, adding the section Update on Correcting this Issue and informing of the correction. There is no necessary update to the associated documentation: https://paritytech.github.io/ink/ink_env/fn.set_contract_storage.html

Testing Guide	In the directory `integration-tests/set_contract_storage` of the target directory, we include in our pull request a test case showing that the observed implementation difference has been corrected. Note that this test is different from the original test case in our repository, which showed the difference.
	
	In order to run the integration tests run:
	`cargo test`
	
	In order to run the e2e tests run:
	`cargo test --features e2e-tests`
	
	These tests are run separately because we do not need to compare their results.

Link to Pull Request	PR2-set-account_storage()

Function	`instantiate_contract()`
Initial Status	Missing Function Implementation on Integration Testing.
Issue Description	There is no implementation of instantiate contract for integration tests. This prevents invoking one contract from another.
Issue Documentation & Test Case	https://github.com/CoinFabrik/on-ink-integration-tests/blob/main/test-cases/instantiate-contract
Current Status	Missing Implementation Performed. Test Cases Passed. Pull Request Performed to Corresponding Repository.

Implementation Notes	In order to allow contract instantiation, we modified the `instantiate_contract()` function to call the dispatch code generated by the ink! code generator. We also added a new `set_contract_storage_test()` function in order to obtain the storage key of the contract, which is now called from the generated code. This function calls `set_contract_storage()` with a specially computed key. This was needed because the code generated for integration tests does not properly generate the storage key for the contract. This way, `instantiate_contract()` can call contract constructors, allowing for the instantiation of contracts in integration tests. Observation: In order to use contract instantiation in integration tests it is necessary to invoke `cargo` with: `cargo test --features test_instantiate` This was necessary because it was not possible to conditionally generate the call to `set_contract_storage_test()` based on whether there is a test being run.
Updated Documentation	We updated the original documentation of this issue in our repo, adding the section Update on Correcting this Issue and informing of the correction. There is no necessary update to the associated documentation: https://paritytech.github.io/ink/ink_env/fn.instantiate_contract.html We did add inline documentation for the new added function `set_contract_storage_test()`.

Testing Guide	In the directory `integration-tests/instantiate-contract` of the target directory, we include in our pull request a test case showing that the observed implementation difference has been corrected. Note that this test is different from the original test case in our repository, which showed the difference.
	
	In order to run the integration tests run:
	`cargo test --features test_instantiate`
	
	In order to run the e2e tests run:
	`cargo test --features e2e-tests`

	These tests are run separately because we do not need to compare their results.
Link to Pull Request	PR2-instantiate-contract()

Ad-hoc developments

As we worked on integrating our test cases into CI/CD, we identified a bug when building e2e tests in a workspace. When contracts are in a workspace with dependencies defined in `Cargo.toml`, and these dependencies are inherited in contracts, the e2e tests fail to compile. However, manually specifying dependencies in each contract resolves the issue. We've logged this bug on GitHub: Issue #1919.

In order to solve Issue #1919, we performed two pull requests:

1. PR in cargo-contract: https://github.com/paritytech/cargo-contract/pull/1358 . In this pull request we initially proposed to add an unstable flag `workspace-mode` that enables the usage of cargo-contracts inside a workspace package. Receiving feedback from the reviewers, they suggested that we test this new functionality against ink-examples (hence the following pull request number 2), and that we make this the default behaviour.
2. PR in ink-examples: https://github.com/paritytech/ink-examples/pull/44 . In this pull request we create an example workspace showing that all examples work correctly with our suggested implementation.

As we worked on rectifying the end-to-end tests to operate within a workspace (Issue #1919), we noticed that this issue addresses a few other issues related to 'cargo-contract' (which also pertains to 'parity'). Namely:

- Add support for workspace inheritance in Cargo.toml paritytech/cargo-contract#1265
- Support multiple contracts in cargo workspace paritytech/cargo-contract#1341
- Fail to build contract with workspace dependencies paritytech/cargo-contract#1172

Further Analysis and Estimation

We analyzed the remaining 13 functions with pending analysis (issue numbers 12 through 24). These functions showed implementations for both environments, so we focused on identifying implementation differences. For each of these functions we provided documentation and test cases in our repository, completing the analysis of the 24 functions exposed for integration and e2e tests.

Methodology

The analysis consisted on:

a. Creating a smart contract example to test the function in integration tests and e2e tests.

b. Writing at least one test case per function per environment.. We chose to label the tests as failing/not-passing when the implementation is incorrect or absent in the Integration environment.

c. Comparing the results of running each test. Showing that the function works as intended in e2e test but missing development or performing in an unintended way for integration tests, or showing the suspected implementation difference.

d. Evaluating the complexity level of each finding, that is, providing an implementation guideline and implementation time estimate.

Results

After documenting and testing the 13 functions with pending analysis, we found behaviour and implementation differences between integration and e2e environments only for functions `weight_to_fee()` and `balance()`. The other 11 functions show no inconsistencies between both environments.

We also observe that, even though the behaviour of `terminate_contract()` is reasonable for integration tests in their current state, where `invoke_contract()` is unimplemented, it is possible that once there is an implementation of `invoke_contract()` it may be necessary to modify `terminate_contract()` in order to prevent the premature termination of the test. We will apply these changes if necessary in our next milestone.

We summarize below our findings and estimations. On top of the Implementation Time Estimate, each implementation should be complemented with additional QA Time (50%* FTE_IMPLEMENTATION) and Documentation Time (25% * FTE_IMPLEMENTATION).

Reference 2: Findings and Estimations

Function	`weight_to_fee()`
Issue Number	14
Status	Implementation Difference.
Issue Description	When calling weight_to_fee() in a contract while performing an integration test we obtain a value of 100. However when we do it while performing an e2e test we obtain a value of 0, no matter the gas provided.
Detailed Documentation & Test Case	https://github.com/CoinFabrik/on-ink-integration-tests/tree/main/test-cases/weight-to-fee
Implementation Idea	We reviewed that the function weight_to_fee() has multiple implementations. It is complex to determine which one of them is responsible for E2E tests because the large size of the runtime makes debugging very slow. This incorrect behaviour of weight_to_fee() in E2E tests is also observed in paritytech/substrate-contracts-node. We will contact the ink! development team to discuss the error and possible corrections.
Complexity Level	Undetermined [TBD in next milestone]
Implementation Time Estimate	Undetermined [TBD in next milestone]

Function	`balance()`
Issue Number	17
Status	Implementation Difference
Issue Description	Initial balance differs in both environments. It is set to 1.000.000 in integration tests and to 1.000.000.000 in e2e tests.
Detailed Documentation & Test Case	https://github.com/CoinFabrik/on-ink-integration-tests/tree/main/test-cases/balance
Implementation Idea	We need to understand where the initial balance value is created/used in the integration test environment and update its value to the value in the e2e environment.

Complexity Level	Feasible
Implementation Time Estimate	3 days [FTE=1]

Conclusions

For the universe of 24 functions in our analysis scope, we update their status and outline the next steps of our work.

Updated Status

We implemented and performed the corresponding pull requests for the functions `1-default_accounts()`, `2-set_contract_storage()` and `7-instantiate_contract()`.

Of the remaining 21 functions:

- Functions `5-gas_left()` and `11-call_runtime()` had been deemed unfeasible for implementation in the analysis performed in our previous grant milestone.
- Our analysis and test cases on functions 12, 13, 15, 16, 18, 19, 20, 21, 22, 23 and 24 showed no implementation differences between integration and e2e testing environments.
- To the list of functions identified in our previous milestone and with implementation plans (`3-invoke_contract_delegate()`, `4-invoke_contract()`, `6-set_code_hash()`, `8-caller_is_origin()`, `9-code_hash()`, `10-own_code_hash()`), we now add function `17-balance()`. We will also continue the review of `weight_to_fee()` and add it to our implementation list if necessary.

Table 3: Updated Status

Issue Number	Function	Implemented Integration Tests	Implemented Integration E2E Tests	Updated Status
1	default_accounts()	Yes	Yes	Implementation Difference Corrected. Pull request performed.
2	set_contract_storage()	Yes	Yes	Missing limitation on Integration Testing Implemented. Pull request performed.

3	invoke_contract_delegate( )	No	Yes	Missing Function Implementation on Integration Testing.
4	invoke_contract()	No	Yes	Missing Function Implementation on Integration Testing
5	gas_left()	No	Yes	Missing Function Implementation on Integration Testing. **Unfeasible Implementation.**
6	set_code_hash()	No	Yes	Missing Function Implementation on Integration Testing.
7	instantiate_contract()	No. Pull request performed.	Yes	Missing Function Implementation on Integration Testing Performed. Pull request performed.
8	caller_is_origin()	No	Yes	Missing Function Implementation on Integration Testing.
9	code_hash()	No	Yes	Missing Function Implementation on Integration Testing.
10	own_code_hash()	No	Yes	Missing Function Implementation on Integration Testing.
11	call_runtime()	No	Yes	Missing Function Implementation on Integration Testing. **Unfeasible Implementation.**
12	caller()	Yes	Yes	Ok. No difference observed in testing.
13	transferred_value()	Yes	Yes	Ok. No difference observed in testing.
14	weight_to_fee()	Yes	Yes	Implementation Difference.

15	block_timestamp()	Yes	Yes	Ok. No difference observed in testing.
16	account_id()	Yes	Yes	Ok. No difference observed in testing.
17	balance()	Yes	Yes	Implementation Difference.
18	block_number()	Yes	Yes	Ok. No difference observed in testing.
19	minimum_balance()	Yes	Yes	Ok. No difference observed in testing.
20	terminate_contract()	Yes	Yes	Ok. No difference observed in testing.
21	transfer()	Yes	Yes	Ok. No difference observed in testing.
22	hash_bytes()	Yes	Yes	Ok. No difference observed in testing.
23	hash_encoded()	Yes	Yes	Ok. No difference observed in testing.
24	ecdsa_recover()	Yes	Yes	Ok. No difference observed in testing.

Next Steps

We will perform any solicited changes related to the pull requests of the implemented functions.

We will continue the estimation and correction of implementation differences for `weight_to_fee()`.

We will perform a new grant application with an implementation proposal for the 7 remaining functions with implementation plans.