Testing the RandomSwap contract

Validation of the Randomness source

What to test	How to test	Expected result	Test performed	Test result
Valid randomness source with correct hash	This is one of the characteristics of the regular valid swap transaction, as we want one token to be swapped for another token. To test is to simply do a regular swap transaction with all the data present	Transaction should be **successful** as long as it spends one token from the contract and offers another valid NFT in return	Successful. Example transaction: https://preview.cardanoscan.io/transaction/178ecc07ddeebe9652e12e7c8d572bef48a3c6b536c54cd7cf4b0e25d07fa49a	
Invalid hash for given slot/blockHash	This can be tested by creating a transaction with a different random hash in the redeemer	Transaction should **fail** because the calculated hash will not match with the expected hash according hash calculated from the slot and blockHash	Failed. The transaction was rejected. The contract responded with: 'Script debugging logs: Hash validation failed\n'	
Manipulated randomIntegerHex not matching random string	This can be tested by creating a transaction with a different random integer hex in the redeemer	Transaction should **fail** because the integer hash will not be contained in the hash created by hashing slot and blockHash	Failed. The transaction was rejected. Response: "Script debugging logs: Random integer hex doesn't match random string\n"	
Incorrect target value derivation from randomIntegerHex	This can be tested by creating a transaction with a random targetValue which is not the same as the expected randomIntegerHex	Transaction should **fail** because the integer hash will calculate into a particular integer and this cannot be any other integer than expected	Failed. The transaction was rejected. Response: 'Script debugging logs: Target value not correctly derived from random hex\n'	

Validation of the Token swap rules

What to test	How to test	Expected result	Test result	Test result as expected
Exactly one token swapped	This is one of the characteristics of the regular valid swap transaction, as we want one token to be swapped for another token. To test is to simply do a regular swap transaction with all the data present	Transaction should be **successful** as long as it spends one token from the contract and offers another valid NFT in return	Successful. Example transaction: https://preview.cardanoscan.io/transaction/536a2803e1ba08bc7cacf268c439d0cae91aebf936d7fac7536f3c2c2619dc36?tab=utxo	
Attempting to swap multiple tokens	We will test this by building a transaction that spends two NFTs from the random swap pool contract and deposits two valid NFTs back	The transaction should **fail**. Contract should allow only one token to be spent and one token to be deposited back to the contract during one transaction. The oracle data should also only qualify one of the requested NFTs to be a valid selection within the selection criterias, because the oracle data will point to only one possible selection per transaction.	Failed. Both tested to withdraw two NFTs from the pool while depositing one and also to withdraw two NFTs from the pool while depositing two. Both tests failed to validate as expected. Two for one:'Script debugging logs: Number of tokens requested not equal to number of tokens received\n' Two for two: 'Script debugging logs: Exactly one token must be swapped\n'	
Attempting to swap zero tokens	Will test this by building a transaction that leaves the NFTs be and only spends ADA from the contacts.	The transaction should **fail**. The contract should only allow transactions that deposits a desired NFT from the queue contract to the swap pool and withdraws a desired NFT from the swap pool which is sent back to the swap initiator	Failed. The transaction failed with the error: 'Script debugging logs: Exactly one token must be swapped\n'	
Attempting to include unlisted tokens	Will test this by building a transaction that deposits an unlisted NFT from the swapper wallet with a desired NFT from the swap pool	The transaction should **fail**. The swap pool should report that there is a difference in the number of desired NFTs deposited in regards to the number of desired NFTs that are withdrawn from the pool	Failed. The transaction failed with the error: 'Script debugging logs: Number of tokens requested not equal to number of tokens received\n'. As a confirmation, the test was also done to see how the swap pool contract sees the NFTs. Result: Counting the number of	

			desired NFTs received: 'Script debugging logs: numDesiredTokensReceived: 0\n' Counting then number of desired NFTs requested: 'Script debugging logs: numDesiredTokensRequested: 1\n'	
Attempting to spend without depositing replacement	Will test this by building a transaction that only withdraws a desired NFT from the swap pool, while not depositing any NFT in return	The transaction should **fail**. The swap pool should report that there is a difference in the number of desired NFTs deposited in regards to the number of desired NFTs that are withdrawn from the pool	Failed. The transaction failed with the error: 'Script debugging logs: Number of tokens requested not equal to number of tokens received\n'. Additional test to count the NFT difference: Result: Counting the number of desired NFTs received (by the pool): 'Script debugging logs: numDesiredTokensReceived: 0\n' Counting the number of desired NFTs requested from the pool: 'Script debugging logs: numDesiredTokensRequested: 1\n'	

Validation of missing or malformed transaction data

What to test	How to test	Expected result	Test result	Test result as expected
Correct reference input with valid datum	This is one of the characteristics of the regular valid swap transaction, as a reference input is required. To test is to simply do a regular swap transaction with all the data present	Transaction should be **successful** to spend the contract UTxO if the datum corresponds with the redeemer data and the name of the NFT being spent	Successful. Example transaction: https://preview.cardanoscan.io/transaction/536a2803e1ba08bc7cacf268c439d0cae91aebf936d7fac7536f3c2c2619dc36?tab=utxo	
Missing reference input	Remove the oracle reference input from the regular transaction	Transaction should **fail** as the oracle is required information for the validation of the selected NFT	Failed. Error message: 'Script debugging logs: fromJust: Nothing\n' could have been more descriptive, but the script did not allow the swap because the UTxO was missing from the transaction	
Invalid datum structure	When there is an invalid datum structure, the validator will get any oracle information and hence not be able to validate the redeemer data. We have UTxOs with Unit (empty) datums in the swap pool, so we can do a transaction and use one as a reference input with non-oracle format.	Transaction should **fail** as the validator will not be able to process the datum structure as expected	Failed. Error message: ‘Script debugging logs: fromJust: Nothing\n’. This is the same error message as for the missing reference input. This makes sense, as the validator is unable to find the datum it looks for in the reference UTxO.	

Edge Cases and Attack Vectors

What to test	How to test	Expected result	Test result	Test result as expected
Very large integers in redeemer fields	This can be tested by creating a redeemer with valid hash bytes, but with very large integer values. I performed the test with the value 18446744073709551615 for all integers other than the action	The validator should **fail** the transaction, but not because of issues with reading the integers, but instead solving the regular validations which will not be correct for all large numbers	Failed. The transaction failed with error: 'Script debugging logs: Hash validation failed\n'	
Zero or negative values in fields	This can be tested by creating a redeemer with valid hash bytes, but with random 0's or negative values for the other integer fields	The validator should **fail** the transaction, but not because of issues with reading the integers, but instead solving the regular validations which will not be correct for all large numbers	Failed. The transaction failed with error: 'Script debugging logs: Hash validation failed\n'	