Quantstamp Security Assessment Certificate

September 7th 2022 — Quantstamp Verified

Lukso LSPs

This audit report was prepared by Quantstamp, the leader in blockchain security.

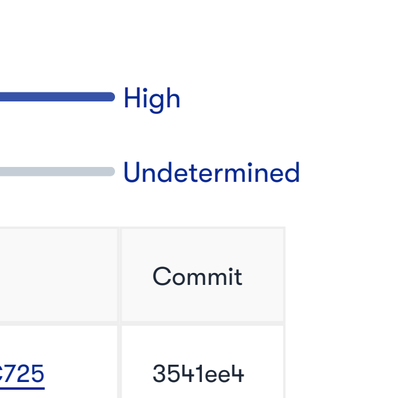
Executive Summary

Type	Industry Standards for Universal Profile and Asset Contracts
Auditors	Poming Lee, Senior Research Engineer Bohan Zhang, Auditing Engineer Rabib Islam, Research Engineer
Timeline	2022-07-11 through 2022-09-08
EVM	Gray Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Public Facing Documents LSP standards
Documentation Quality	High
Test Quality	Undetermined

Source Code

Repository	Commit
ERC725Alliance/ERC725	3541ee4
lukso-network/lsp-smart-contracts	602b79b
ERC725Alliance/ERC725	5c47854
lukso-network/lsp-smart-contracts	3f14866

Total Issues	**14** (8 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**6** (5 Resolved)
Informational Risk Issues	**8** (3 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Quantstamp has done the audit for Lukso's LSPs, on a best-effort basis, with 3 auditors working independently (based on a commit-reveal scheme and later on syncing on their findings). During the course of this audit, we have found a few issues with low and informational severity levels. Overall, the code and the documentation appear to be of high quality; however, the coverage score of the test suite cannot be determined at this moment due to the testing framework used. The dev team has been very professional and helpful throughout the course of the audit. Although we have discovered some issues that may be fixed before the project goes live, we believe that the project may remain risky in the sense that as an industry standard, the project is novel, contains intricate logic, and is expected to be extensively applied to a wide variety of user scenarios with/without being modified by different developers from other projects. Unexpected new ways of using the standards may arise and edge cases are extremely difficult to predict. We highly recommend extensively testing the code (at least improving all coverage to >90%) and thinking of more edge cases before going live. Also, continuous on-chain observation and revision of the standards based on how people/projects are using them are recommended.

2022-09-08 Update: During this re-audit, the admin team has brought all the status of findings into fixed, acknowledged, or mitigated. The coverage scores have been obtained. At this moment the branch coverage for `ERC725Alliance/ERC725` is 88.1% and 82.23% for `lukso-network/lsp-smart-contracts`.

ID	Description	Severity	Status
QSP-1	Overwriting without User Acknowledgement Due to Hash Collision/Mistake	Low	Acknowledged
QSP-2	Operator Could Clear Operator List	Low	Fixed
QSP-3	Missing Input Validation	Low	Fixed
QSP-4	Missing `_disableInitializers` for some Contracts	Low	Fixed
QSP-5	`LSP6KeyManagerCore::_countTrailingZeroBytes()` Can Not Recognize the Condition Where `key == 0`	Low	Fixed
QSP-6	Incorrect Key Prefix Could Be Used	Low	Fixed
QSP-7	Critical Functions Are Not Protected From Reentrancy	Informational	Acknowledged
QSP-8	On Potential Hash Collisions	Informational	Acknowledged
QSP-9	Allowance Double-Spend Exploit	Informational	Acknowledged
QSP-10	Repeatedly Adding the Same Operator for One TokenID	Informational	Fixed
QSP-11	Unlocked Pragma	Informational	Acknowledged
QSP-12	Possible to Renounce Ownership	Informational	Mitigated
QSP-13	`LSP1UniversalReceiver` Can Lead to Unexpected Reentrancy Behavior	Informational	Acknowledged
QSP-14	Remove Non-Existing Operator	Informational	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

DISCLAIMER:
If the final commit hash provided by the client contains features that are not in scope of the audit or a re-audit, those features are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Overwriting without User Acknowledgement Due to Hash Collision/Mistake

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `ERC725-3541ee4/implementations/contracts/ERC725YCore.sol`

Description: Users may call `setData()` with an existing key by mistake or because of a hash collision. As the core part of this project. It is suggested to distinguish two modes, one of which sets new data, while the other modifies existing data.

Recommendation: Consider adding relative checks when calling `ERC725YCore::_setData()`. For instance, by adding a `boolean` to API representing if users allow an overwritten.

Update: The dev team stated:

```
ERC725Y is a general module that allows mapping data keys to data values. These data keys are not checked by default if they are mapped to some data value or not.
To check if a data key is mapped to some value or not before writing is seen as an optional feature for some users implementing the ERC725Y standard. This design decision is also supported by the additional gas cost that
would be required to read the contract storage to check for overriding or not.
If a check should be added to avoid overwriting existing entries, it should be added on the LSP6KeyManager standard instead of on LSP0. An issue was opened on the LIPs repo to discuss splitting SETDATA Permission into
```

```
EDITDATA and ADDDATA Permissions.
See [issue #110](https://github.com/lukso-network/LIPs/issues/110) in lukso-network/LIPs repository.
```

QSP-2 Operator Could Clear Operator List

Severity: *Low Risk*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol`

Description: For the function `revokeOperator()`, the dev team only allows the token owner to revoke the operator. However, if any operator calls `transfer(from = owner, to = owner)`, then the operator list is cleared, and the owner will still have the NFT. The effect would be equivalent to that the operator successfully revokes all operators.

Recommendation: Re-consider whether this behavior is allowed from the standards' perspective. Otherwise, add relevant checks to stop this unexpected edge case from occurring.

Update: The dev team stated:

```
A custom error in LSP8 was introduced to revert when the from and to addresses are the same, to avoid this unexpected side effect of clearing the list of operators for a specific tokenId.
See [PR #266](https://github.com/lukso-network/lsp-smart-contracts/pull/266) in lukso-network/lsp-smart-contracts repository.
```

QSP-3 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/Custom/ClaimOwnership.sol`

Description: To avoid human error, it is important to validate inputs even if they only come from trusted addresses. The following functions do not have a proper validation of input parameters:

1. `lsp-smart-contracts-602b79b/contracts/Custom/ClaimOwnership.sol::transferOwnership()` does not check if `newOwner` is `0x0`.
2. `lsp-smart-contracts-602b79b/contracts/LSP4DigitalAssetMetadata/LSP4DigitalAssetMetadata.sol::constructor()` does not check if `newOwner_` is `0x0`.
3. `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol::_burn()` does not check if the NFT token it wants to burn exists or not.

Recommendation: Consider adding relevant checks.

Update: The dev team stated:

```
Regarding the missing input validations, below are the details of actions taken on each points.

1.    lsp-smart-contracts-602b79b/contracts/Custom/ClaimOwnership.sol::transferOwnership() does not check if newOwner is 0x0.

Not implemented in the ClaimOwnership contract as in some cases, transferring ownership to the address(0) can be relevant. Examples:
-    to reset the pendingOwner to address(0) if the owner made a mistake in the address provided as a parameter.
-    if the owner changes his mind and does not want to transfer ownership anymore.

2.    lsp-smart-contracts-602b79b/contracts/LSP4DigitalAssetMetadata/LSP4DigitalAssetMetadata.sol::constructor() does not check if newOwner_ is 0x0.

Fixed and implemented directly in:
-    LSP4DigitalAssetMetadata.sol and LSP4DigitalAssetMetadataInitAbstract.sol: [PR #268](https://github.com/lukso-network/lsp-smart-contracts/pull/268) on lukso-network/lsp-smart-contracts repository.
-    ERC725, ERC725InitAbstract, ERC725X, ERC725XInitAbstract, ERC725Y and ERC725YInitAbstract: [PR #159](https://github.com/ERC725Alliance/ERC725/pull/159) in ERC725Alliance/ERC725 repository.

A new 3.1.3 version of the @erc725/smart-contracts package will be released and the dependency will be upgraded in the @lukso/lsp-smart-contracts repository to ensure that this requirement check is fixed on all the LSP smart
contracts implementations that inherit from any ERC725 contracts.

3.    lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol::_burn() does not check if the NFT token it wants to burn exists or not.

Not applicable: the internal _burn(...) function checks if the tokenId passed as a parameter exists through the function tokenOwnerOf(...). See [LSP8DigitalAssetCore.sol, line 307](https://github.com/lukso-network/lsp-smart-
contracts/blob/4576e9dd6bf7597ac06d8036a19e817cc35611c7/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol#L307) and [LSP8DigitalAssetCore.sol, lines 69-71](https://github.com/lukso-network/lsp-
smart-contracts/blob/4576e9dd6bf7597ac06d8036a19e817cc35611c7/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol#L69-L71).
More tests were added in commit 4576e9d ([PR #268](https://github.com/lukso-network/lsp-smart-contracts/pull/268)) to ensure that the _burn(...) function reverts when the tokenId given as a parameter does not exist.
```

QSP-4 Missing `_disableInitializers` for some Contracts

Severity: *Low Risk*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP4DigitalAssetMetadata/LSP4DigitalAssetMetadataInit.sol`, `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/extensions/LSP7CappedSupplyInit.sol`, `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/presets/LSP7CompatibleERC20MintableInit.sol`, `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721Init.sol`, `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/presets/LSP8CompatibleERC721MintableInit.sol`, `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupplyInit.sol`

Description: An uninitialized contract can be taken over by an attacker. This applies to both a proxy and its implementation contract, which may impact the proxy. To prevent the implementation contract from being used, you should invoke the `_disableInitializers` function in the constructor to automatically lock it when it is deployed.
The following contracts' `constructor` is not protected by `_disableInitializers`:

1. `LSP4DigitalAssetMetadataInit`
2. `LSP7CappedSupplyInit`
3. `LSP7CompatibleERC20MintableInit`
4. `LSP8CompatibleERC721Init`
5. `LSP8CompatibleERC721MintableInit`
6. `LSP8CappedSupplyInit`

Recommendation: Consider adding `_disableInitializers` to the `constructor` of the listed contracts to enhance security.

Update: The dev team stated:

```
The following changes were applied to resolve this issue:
```

```
-    A constructor with a call to _disableInitializers(...) was added in the following contracts
    -   LSP7CompatibleERC20MintableInit
    -   LSP8CompatibleERC721MintableInit

-    The following contracts from the extensions/ folder of LSP7 and LSP8 were removed.
    -   LSP4DigitalAssetMetadataInit
    -   LSP7CappedSupplyInit
    -   LSP8CappedSupplyInit
    -   LSP7CompatibleERC20Init
    -   LSP8CompatibleERC721Init

These contracts are token extension and should be used via inheritance, not deployed directly. The inheritance of other contracts was changed to use the InitAbstract contracts in replacement.
See [PR #265](https://github.com/lukso-network/lsp-smart-contracts/pull/265) in lukso-network/lsp-smart-contracts repository:
```

QSP-5 `LSP6KeyManagerCore::_countTrailingZeroBytes()` Can Not Recognize the Condition Where `key == 0`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol`

Description: On `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L616`, the minimum `index` is `0`, so the max value of the function output result would be `31` instead of `32`.

Recommendation: Consider changing the while loop condition into `while (index >= 0 && key[index] == 0x00) index--;` to fix this edge case.

Update: The dev team stated:

```
This issue was found by the internal smart contract team during the audit. There was an underflow in the function _countTrailingZeroBytes(...) if one of the allowed ERC725Y Keys in the list was bytes32(0).

See [PR #226](https://github.com/lukso-network/lsp-smart-contracts/pull/226) in lukso-network/lsp-smart-contracts where the issue was initially resolved.

An extra bug was also found in the same function. The index in the while loop was checked incorrectly. The function was refactored with more tests were added to ensure the correct behaviour (for instance, when an allowed
ERC725Y contains only 1 bytes, and 31 trailing zero bytes).

See [PR #264](https://github.com/lukso-network/lsp-smart-contracts/pull/264) in lukso-network/lsp-smart-contracts.
```

QSP-6 Incorrect Key Prefix Could Be Used

Severity: *Low Risk*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol`

Description: According to the documentation, data keys of type "Mapping" and "MappingWithGrouping" are formatted such that there are two empty bytes between the first ten bytes and the last twenty bytes. However, the versions of the functions `generateMappingKey` and `generateMappingWithGroupingKey` that use byte inputs take `bytes12` for the first part of the key without ensuring that there are two empty bytes as per the specification.

Recommendation: Consider making the functions take `bytes10` for the first argument, and while generating the data key, make sure to concatenate two empty bytes in the middle of the key.

Update: The dev team stated:

```
All the literal constants for LSP2 key types Mapping and MappingWithGrouping were edited to remove the last two zero bytes. Both functions were refactored as per the suggested changes to append two zero bytes at the end.

See [PR #269](https://github.com/lukso-network/lsp-smart-contracts/pull/269/files) in lukso-network/lsp-smart-contracts.
```

QSP-7 Critical Functions Are Not Protected From Reentrancy

Severity: *Informational*

Status: Acknowledged

File(s) affected: `ERC725-3541ee4/implementations/contracts/ERC725XCore.sol`, `lsp-smart-contracts-602b79b/contracts/LSP0ERC725Account/LSP0ERC725AccountCore.sol`, `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol`, `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/LSP7DigitalAssetCore.sol`, `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol`, `lsp-smart-contracts-602b79b/contracts/LSP9Vault/LSP9VaultCore.sol`

Description: There are several critical functions in the system that are not protected from re-entrancy and this pattern could be used as a tool to conduct a complex attack that involves re-entrancy attack.

1. `ERC725XCore::execute()`
2. `LSP0ERC725AccountCore::universalReceiver()`
3. `LSP6KeyManagerCore::executeRelayCall()`
4. `LSP6KeyManagerCore::execute()`
5. `LSP7DigitalAssetCore.sol::_transfer()`
6. `LSP7DigitalAssetCore.sol::_mint()`
7. `LSP7DigitalAssetCore.sol::_burn()`
8. `LSP8IdentifiableDigitalAssetCore::_transfer()`
9. `LSP8IdentifiableDigitalAssetCore::_mint()`
10. `LSP8IdentifiableDigitalAssetCore::_burn()`
11. `LSP9VaultCore::universalReceiver()`

Recommendation: It is highly recommended to inherit (ReentrancyGuard.sol)[https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol] and add `nonReentrant` modifier to all the critical functions that are not expected to be re-entered.

Update: The dev team stated:

```
1.    ERC725XCore::execute(): Will NOT receive an reentrancy guard, as its a core module, to be used in a variety of contracts. Reentrancy is a feature here.
```

```
2.    LSP0ERC725AccountCore::universalReceiver(): Will NOT receive an reentrancy guard, as its a feature to notify the UP of multiple actions that happen during a transaction. For example sending of batch transactions via a LSP7/8 transferBatch() call.
3.    LSP6KeyManagerCore::executeRelayCall()
LSP6KeyManagerCore::execute(): Will ADD an reentrancy guard in the current implementation, to keep the risk smaller. With the goal to remove it in the future, when risks and use cases are better understood, as reentrancy is a feature for potential use cases here, and the KM safeguards entrance based on permission.
4.    LSP7DigitalAssetCore.sol::_transfer(): Will NOT receive an reentrancy guard as its a feature for protocols and its protected as the msg.sender needs to have a balance.
5.    LSP7DigitalAssetCore.sol::_mint(): Will NOT receive an reentrancy guard on the internal function, its a feature for protocols and its protected as the msg.sender is the owner in the public mint(...) function of the preset contracts.
6.    LSP7DigitalAssetCore.sol::_burn(): Will NOT receive an reentrancy guard on the internal function, its a feature for protocols and its protected as the msg.sender is the owner.
7.    LSP8IdentifiableDigitalAssetCore::_transfer(): Will NOT receive an reentrancy guard as its a feature for protocols and its protected as the msg.sender needs to have a balance.
8.    LSP8IdentifiableDigitalAssetCore::_mint(): Will NOT receive a reentrancy guard on the internal function, its a feature for protocols and its protected as the msg.sender is the owner in the public mint(...) function of the preset contracts.
9.    LSP8IdentifiableDigitalAssetCore::_burn(): Will NOT receive an reentrancy guard on the internal function, its a feature for protocols and its protected as the msg.sender is the owner.
10.    LSP9VaultCore::universalReceiver(): Will NOT receive an reentrancy guard, as its a feature to notify the Vault of multiple actions that happen during a transaction. For example sending of batch transactions via a LSP7/8 transferBatch() call.
See [issue #286](https://github.com/lukso-network/lsp-smart-contracts/issues/286) in lukso-network/lsp-smart-contracts repository.
```

Notably, point (3) has not been added to the code base at this moment when we do the fix review.

QSP-8 On Potential Hash Collisions

Severity: *Informational*

Status: Acknowledged

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol`

Description: In the function `generateMappingWithGroupingKey(firstWord, secondWord, addr)`, when `firstword` and `addr` are the same, there is a `0.0000000233%` possibility that two random variables `secondWord` will result in the same key.

Recommendation: Consider adding this information to the public-facing documentation to make sure that users and devs working on related projects are aware of this risk and would be taking action to avoid it. Another approach to prevent this collision from happening is to explicitly reject the condition where `firstWord == addr`.

Update: The dev team stated:

```
A warning of potential hash collision for LSP2 key type MappingWithGrouping was documented in docs.lukso.tech (Standards > Generic Standards > LSP2 > MappingWithGrouping.

See [PR #369](https://github.com/lukso-network/docs/pull/369) in lukso-network/docs.
```

QSP-9 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Acknowledged

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/LSP7DigitalAssetCore.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20/LSP7 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `authorizeOperator()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `authorizeOperator()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transfer()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain the ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls the`transfer()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) can be mitigated through the use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on a traditional `approve()` / `transferFrom()` should keep in mind that they have to set allowance to `0` first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contracts.

Update: The dev team stated:

```
As a secure flow exists due to setting the allowance to 0 and the setting it to the wanted number, we don't see a strong need for extra functions in the standard.
We will discuss in the future within the community if it makes sense to add two new functions to the standard.

See the following links for references:
-    [Issue #112](https://github.com/lukso-network/LIPs/issues/112) in lukso-network/LIPs repository.
-    [PR #277](https://github.com/lukso-network/lsp-smart-contracts/pull/277) in lukso-network/lsp-smart-contracts
-    [PR #373](https://github.com/lukso-network/docs/pull/373) in lukso-network/docs
-    [PR #111](https://github.com/lukso-network/LIPs/pull/111) in lukso-network/LIP repository
```

QSP-10 Repeatedly Adding the Same Operator for One TokenID

Severity: *Informational*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol`

Description: Users can call the function `authorizeOperator()`, multiple times with the same arguments. That will also cause the same event to be emitted multiple times.

Recommendation: Consider checking the returned `boolean` of `EnumerableSet::add()` in `authorizeOperator()`.

Update: The dev team stated:

```
The function authorizeOperator(...) in LSP8 was refactored to revert when an operator for a specific tokenId is already approved.

See [PR #270](https://github.com/lukso-network/lsp-smart-contracts/pull/270) in lukso-network/lsp-smart-contracts
```

QSP-11 Unlocked Pragma

Severity: *Informational*

Status: Acknowledged

File(s) affected: `ERC725-3541ee4/*,lsp-smart-contracts-602b79b/*`

Description: Every Solidity file specifies in the header a version number of the format pragma solidity (^)0.8.*. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version and above, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

Update: The dev team stated:

```
The SWC-103 (Smart Contract Weakness Classification nb 103) mentions:

“Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.”

This is the case of the lsp-smart-contracts. These contracts are intended to be used as a library and consumed by other developers. Therefore, a floating pragma was kept on all the contracts.

See [PR #275](https://github.com/lukso-network/lsp-smart-contracts/pull/275) in lukso-network/lsp-smart-contracts repository.
```

QSP-12 Possible to Renounce Ownership

Severity: *Informational*

Status: Mitigated

File(s) affected: `ERC725-3541ee4/implementations/contracts/custom/OwnableUnset.sol`

Description: Contract `OwnableUnset` contains the function `renounceOwnership`, which, if called, would render the contract largely unusable. In particular, it would leave an `LSP0ERC725Account` without `KeyManager`, resulting in much of the contracts' functionality being unusable. This could lead to loss of funds and in some cases break composability where the contract in concern is part of a greater system of contracts.

Recommendation: Override the `renounceOwnership` function in contracts that should not be left without an owner.

Update: The dev team stated:

```
RenounceOwnership is a valid use case when UP owners die, or NFTs should be finally locked.

-    In ERC725 contracts RenounceOwnership is left in its basic form, as those are module contracts for a variety of other smart contracts.

-    For LSP0 and LSP9 we changed it now to two step process, where the the user calls renounceOwnership(), which starts a time in blocks, where for the first 100 blocks, he can not complete the renouncement, and after 100 blocks he has the ability to call renounceOwnership() again to complete the renouncement. If after 200 blocks after the initial call to renounceOwnership() the process was not completed, the user will have to start all over again.
We also added a RenounceOwnershipInitiated Event to notify the user of a started process.

This will allow for a delay time for the owner to act, if a malicious controller initiated the process. And allows for an expiry, the process is not completed within 100 blocks (~20min)

-    The LSP6 implementation will not allow renounceOwnership(...) at all for now.

See the following references for more details:
-    [Issue #115](https://github.com/lukso-network/LIPs/issues/115) in lukso-network/LIPs.
-    [PR #282](https://github.com/lukso-network/lsp-smart-contracts/pull/282) in lukso-network/lsp-smart-contracts.
```

QSP-13 `LSP1UniversalReceiver` Can Lead to Unexpected Reentrancy Behavior

Severity: *Informational*

Status: Acknowledged

Description: In scenarios where permissions are set in certain specific ways, users may trigger transactions that cause their balances to change much more drastically than expected. Consider the following scenario:

1. Alice authorizes Bob as an operator for `5000` tokens.
2. Bob uses a universal receiver with logic that calls `transfer` using the sender's funds to send the remainder of the sender's balance to himself.
3. Alice transfers `200` tokens to Bob.
4. Bob receives `5000` tokens, more than Alice expected.

Although Alice did authorize Bob to transfer her tokens as she saw fit, she did not expect that merely transferring `200` tokens would wipe out her entire balance.

Recommendation: Properly notify users of the potential for unexpected behavior due to `universalReceiver` when functions like `authorizeOperator` are called.

Update: The dev team stated:

```
We introduced a warning in the LSP7 and LSP8 pages on docs.lukso.tech.

Regarding the example documented in the audit report, we do not consider this as a security issue in this context. Since Bob was granted an operator allowance in the first place by Alice, he could also transfer all the tokens to himself in a separate transaction.

It should be up to the LSP7 token contract implementation to decide if this behaviour should be allowed or not by adding a reentrancy guard on the transfer(...) function. It is something to not restrict on the standard level or the implementation.

Since the LSP1UniversalReceiverDelegate is the only gateway for users to react depending on the action they get notified about (token transfers, etc.), there could be use cases where the universalReceiver(...) function should be reentered to perform specific actions.

A notice was added in docs.lukso.tech about external calls being done via hooks when LSP7 and LSP8 assets are transferred.

See [PR #376](https://github.com/lukso-network/docs/pull/376) in lukso-network/docs repository for more details.
```

QSP-14 Remove Non-Existing Operator

Severity: *Informational*

Status: Fixed

File(s) affected: `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol`

Description: When the user calls `_revokeOperator()` with a non-existing operator, the function will terminate successfully and with an incorrectly emitted event.

Recommendation: Consider checking the returned `boolean` of `EnumerableSet::add()` in `_revokeOperator()`.

Update: The dev team stated:

```
The function revokeOperator(...) in LSP8 now revert when the caller tries to revoke an operator that does not exist, or is not authorised as an operator for a specific tokenId.

See [PR #271](https://github.com/lukso-network/lsp-smart-contracts/pull/271) in lukso-network/lsp-smart-contracts repository.
```

Automated Analyses

Slither

Slither reported `587` results, all of which were either identified as false positives or included in the findings of this report.

Code Documentation

1. claimownership: a typo "In EIP713", should be "In EIP173" instead.
2. On `L140`, `L157`, and `L177` of `ERC725-3541ee4/implementations/contracts/ERC725XCore.sol`, "Unknow Error" should be "Unknown Error"
3. On `L107`, `L148` of `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol`, "Unknow Error" should be "Unknown Error"
4. On `L145` of `lsp-smart-contracts-602b79b/contracts/LSP5ReceivedAssets/LSP5Utils.sol`, "elemnt" should be "element"
5. `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/ILSP6KeyManager.sol::L36`: `_calldata` -> `payload`
6. `lsp-smart-contracts-602b79b/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol`: Not every function has NatSpec.
7. In LSP6: "they check on restrictions for addresses, standards, or functions are skipped." is improper grammar.
8. In LSP6: similarly, "This allows for cheaper transactions where, these restrictions aren't set anyway."
9. In LSP6: sequentiel -> sequential
10. In LSP6: sequentielially -> sequentially
11. singleton, the "key" inside the example JSON:

```
{
  "name": "SupportedStandards:LSP3UniversalProfile",
  "key": "0xeafec4d89fa9619884b6b89135626455000000000000000000000000abe425d6",
  "keyType": "Mapping",
  "valueType": "bytes4",
  "valueContent": "0xabe425d6"
}
```

is different from the key listed in the image. Please check which one is correct.

======2022-09-08 Update: all fixed======

Adherence to Best Practices

1. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/Factories/UniversalFactory.sol::L135`.
2. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/LSP0ERC725Account/LSP0ERC725AccountCore.sol::L60`.
3. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/LSP1UniversalReceiverDelegateUP.sol::L63`
4. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L453`
5. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L644`
6. [fixed] TODO on `lsp-smart-contracts-602b79b/contracts/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol::L229`
7. [ack] Functions `_mint` in `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/LSP7DigitalAssetCore.sol` could be re-entered. It is highly recommended to inherit (ReentrancyGuard.sol)[https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol] and add `nonReentrant` modifier to them.
8. [fixed] `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::_verifyCanSetPermissions()`: consider blocking the `else` case by reverting the transaction to prevent edge cases from bypassing the permission checks.
9. [fixed] `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L395`: consider reverting the transaction if the `allowedERC725YKeysEncoded` data is found corrupted.
10. [fixed] `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L549`: consider reverting the transaction if the `allowedAddresses` data is found corrupted.
11. [fixed] `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L574`: consider reverting the transaction if the `allowedStandards` data is found corrupted.
12. [fixed] `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol::L600`: consider reverting the transaction if the `allowedFunctions` data is found corrupted.
13. `lsp-smart-contracts-602b79b/contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/LSP1UniversalReceiverDelegateUP.sol::universalReceiverDelegate()`: unused function parameters: `uint256 value` and `bytes memory data`.
14. `lsp-smart-contracts-602b79b/contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/LSP1UniversalReceiverDelegateVault.sol::universalReceiverDelegate()`: unused function parameters: `uint256 value` and `bytes memory data`.
15. `lsp-smart-contracts-602b79b/contracts/Legacy/UniversalReceiverAddressStore.sol::universalReceiverDelegate()`: unused function parameters: `uint256 value` and `bytes memory data`.
16. [fixed] The import of `Initializable` contract has two distinct sources, they are: "@erc725/smart-contracts/contracts/custom/Initializable.sol" and

"@openzeppelin/contracts/proxy/utils/Initializable.sol". Consider removing one of the sources and leaving only one version of `Initializable` contract to avoid the unexpected condition.

17. [fixed] In `lsp-smart-contracts-602b79b/contracts/LSP0ERC725Account/LSP0ERC725AccountCore.sol`, the function `transferOwnership()` seems to be redundent since it was implemented in `ClaimOwnership.sol` with same name and functionality.
18. [fixed] In `lsp-smart-contracts-602b79b/contracts/LSP7DigitalAsset/LSP7DigitalAssetCore.sol`, function name `isOperatorFor` implies the return type should be a boolean . However, the return value is type `uint`. Consider renaming it similar as `authorizedAmount()` to avoid confusion.
19. [fixed] In `lsp-smart-contracts-602b79b/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol`, function `_requirePermissions()` does not return anything. Consider removing `returns (bool)`.
20. [fixed] ERC725 uses `uncheckedIncrement` for iterating through for loops. However, the LSPs repo does not use this. Consider using the same function for the LSPs to save gas.

======2022-09-08 Update======

For `BP-4`, the dev team stated:

```
DELEGATECALL is dangerous by nature and its effects can hardly be anticipated or mitigated. There might be multiple ways to use DELEGATECALL for the same outcome or to override specific parts of the contract storage.

DELEGATECALL will remain disallowed via the KeyManager. Developers who want to use a UP without a Key Manager should be cautious when using DELEGATECALL and ensure the address being called is trusted and will not have dangerous side effects on the contract storage.
```

For `BP-7`, the dev team stated:

```
No reentrancy guard was applied in the internal _mint(...) function. It should be up to the contract implementing LSP7 to add the reentrancy guard depending on the feature they want to allow or not (see also QSP-1 for additional information).
```

For `BP-9` to BP-12` the code is unchanged. The dev team stated:

```
In the current implementation of LSP6KeyManager, for the following data keys:

-    AddressPermissions:AllowedAddresses:<address>
-    AddressPermissions:AllowedFunctions:<address>
-    AddressPermissions:AllowedStandards:<address>
-    AddressPermissions:AllowedERC725YKeys:<address>

A check is currently applied in two scenarios inside the code of LSP6KeyManagerCore.sol

Scenario 1: When the Key Manager reads the UP to retrieve the list of allowed addresses/functions/standards/ERC725YKeys to perform the permissions checks.

When reading the value stored under these data keys, if the underlying bytes stored are corrupted (not a correctly abi-encoded array), then we consider it like if nothing is set under these data keys. This is equivalent to allowing any address, function, etc… according to LSP6. This is the reason why the code returns directly in the related functions.

The reason for this is because if the data under one of these data keys is corrupted, [abi.decode(...)](https://github.com/lukso-network/lsp-smart-contracts/blob/d608479df753087b33897586f2fe92b90252c869/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L580) will fail and revert. This would lead a controller address with some incorrectly/corrupted data stored under its allowed key to be stuck. The controller address would not be able to do anything (setData(...) or execute(...)) anymore, as every time it would try to call the Key Manager, it would revert on abi-decode(...). This would make its permission SETDATA or CALL unusable.


Scenario 2: when writing data on the UP via the Key Manager, giving some specific allowed addresses/functions/standards/ERC725YKeys for a controller address (= an address with some permissions) as an input.


A check is made on the values given as input when the payload is setData(...) to verify that no corrupted data will be stored for these data keys. This ensure that the potential issue mentioned in scenario 1 (a controller being stuck) does not occur in the first place.

Therefore the values given as input for setting the AllowedAddresses, AllowedFunctions, AllowedStandards or AllowedERC725YKeys are checked to ensure they are correctly abi-encoded arrays. If they are not, we [revert execution](https://github.com/lukso-network/lsp-smart-contracts/blob/d608479df753087b33897586f2fe92b90252c869/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L317-L319).

To conclude, we have these checks in both places (scenario 1 and scenario 2) because these data keys can be set before a Key Manager becomes the owner of an LSP0/Universal Profile. This way, we can catch the potential bug (of a controller address being stuck) by anticipation (scenario 1), and prevent the bug from happening afterwards (scenario 2).
```

For `BP-16`, the dev team stated:

```
The Initializable contract from @erc725/smart-contracts has been removed. Once a new 3.1.3 version of the @erc725/smart-contracts package is released, the dependency will be updated in @lukso/lsp-smart-contracts to fix the error.
```

For `BP-17`, the dev team stated:

```
This function cannot be removed from LSP0ERC725AccountCore.sol because of the inheritance.

This function is also defined in OwnableUnset and needs to be overridden in LSP0. Otherwise, the Solidity compiler does not know which function to refer to (the one from OwnableUnset or from ClaimOwnership) and report the following error:

'Derived contract must override function "transferOwnership". Two or more base classes define function with same name and parameter types'.
```

Test Results

Test Suite Results

All tests have passed.

```
==========ERC725Alliance/ERC725:

Calculate ERC725 InterfaceIDs
 ✓ ERC725X
 ✓ ERC725Y

ERC725
when using ERC725 with constructor
  when deploying the contract
     ✓ should revert when giving address(0) as owner (54ms)
when using ERC725 with proxy
  when deploying the base implementation contract
     ✓ prevent any address from calling the initialize(...) function on the implementation (131ms)
  when deploying the contract as proxy
     ✓ should revert when initializing with address(0) as owner

ERC725X
when using ERC725X contract with constructor
  when deploying the contract
     ✓ should revert when giving address(0) as owner
    once the contract was deployed
      when the contract was initialized
         ✓ should have registered the ERC165 interface
         ✓ should have registered the ERC725X interface
         ✓ should have set the correct owner
  when testing deployed contract
    When testing ownership
      When owner is transferring ownership
         ✓ should pass and emit OwnershipTransferred event
      When non-owner is transferring ownership
         ✓ should revert
      When owner is renouncing ownership
         ✓ should pass and emit OwnershipTransferred event (79ms)
      When non-owner is renouncing ownership
         ✓ should revert
    When testing execution
      When testing execution ownership
        When owner is executing
           ✓ should pass and emit Executed event (45ms)
```

```
      When non-owner is executing
         ✓ should revert
    When testing Operation CALL
        when updating an existing data Key
           ✓ should pass and emit DataChanged event (62ms)
        when removing an existing data Key
           ✓ should pass and emit DataChanged event (57ms)
        when setting large bytes in the storage
           ✓ should pass and emit DataChanged event (1738ms)
    When using setData(bytes32[],bytes[])
      When owner is setting data
         ✓ should pass and emit DataChanged event (78ms)
      When non-owner is setting data
         ✓ should revert
      When interacting from a smart Contract
         ✓ should pass and emit DataChanged event (105ms)
      When interacting from an EOA
        when setting a new data Key
           ✓ should pass and emit DataChanged event (77ms)
        when updating an existing data Key
           ✓ should pass and emit DataChanged event (77ms)
        when removing an existing data Key
           ✓ should pass and emit DataChanged event (74ms)
        when setting large bytes in the storage
           ✓ should pass and emit DataChanged event (1761ms)
        when data keys length != data values length
           ✓ should revert
  When testing getting data
    When using getData(bytes32)
      When owner is setting data
         ✓ should pass
      When non-owner is setting data
         ✓ should pass
      When interacting from a smart Contract
         ✓ should pass
      When interacting from an EOA
        When getting normal sizeData
           ✓ should pass
        When getting big sizeData
           ✓ should pass (80ms)
    When using getData(bytes32[])
      When owner is setting data
         ✓ should pass
      When non-owner is setting data
         ✓ should pass
      When interacting from a smart Contract
         ✓ should pass
      When interacting from an EOA
        When getting normal sizeData
           ✓ should pass
        When getting big sizeData
           ✓ should pass (78ms)
        When getting many entries
           ✓ should pass


187 passing (35s)


==========lukso-network/lsp-smart-contracts:


> @lukso/lsp-smart-contracts@0.6.2 test:parallel
> run-p test:helpers test:up test:lsp1 test:lsp2 test:lsp4 test:lsp6 test:lsp7 test:lsp8 test:lsp9 test:universalfactory


> @lukso/lsp-smart-contracts@0.6.2 test:lsp2
> hardhat test --no-compile tests/LSP2ERC725YJSONSchema/LSP2UtilsLibrary.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp1
> hardhat test --no-compile tests/LSP1UniversalReceiver/*.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:up
> hardhat test --no-compile tests/UniversalProfile.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp6
> hardhat test --no-compile tests/LSP6KeyManager/LSP6KeyManager.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:helpers
> hardhat test --no-compile tests/Helpers/*.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp8
> hardhat test --no-compile tests/LSP8IdentifiableDigitalAsset/*.test.ts tests/LSP8IdentifiableDigitalAsset/extensions/*.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp4
> hardhat test --no-compile tests/LSP4DigitalAssetMetadata/LSP4Compatibility.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp9
> hardhat test --no-compile tests/LSP9Vault/*.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:lsp7
> hardhat test --no-compile tests/LSP7DigitalAsset/*.test.ts tests/LSP7DigitalAsset/extensions/*.test.ts


> @lukso/lsp-smart-contracts@0.6.2 test:universalfactory
> hardhat test --no-compile tests/Factories/UniversalFactory.test.ts


  LSP2Utils
    isEncodedArray(...)
      testing different zero bytes 00 of various length

  LSP4Compatibility
    when using LSP4Compatibility

  UniversalFactory contract
    When Using UniversalFactory

  UniversalProfile
    when using UniversalProfile contract with constructor
      when deploying the contract with or without value

  Address Registry contracts

  LSP9Vault
    when using LSP9Vault contract with constructor
      when deploying the contract
        when initializing the contract
          when the contract was initialized

  LSP1UniversalReceiverDelegateUP
    when testing deployed contract
      when deploying the contract
        when initializing the contract
          when the contract was initialized
         should return false for 1 x empty zero bytes
         should return false for 10 x empty zero bytes
         should return false for 20 x empty zero bytes
       should allow reading name
         should return false for 30 x empty zero bytes
         should return true for 32 x empty zero bytes
       should allow reading symbol

·-----------------------|---------------------------|--------------|-----------------------------·
|  Solc version: 0.8.10  ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
·····························|···························|··············|·····························
|  Methods
·····························|···························|··············|·····························
|  Contract   ·  Method  ·  Min        ·  Max        ·  Avg         ·  # calls    ·  usd (avg)
·-------------|----------|-------------|-------------|--------------|-------------|--------------·

  2 passing (31s)
```

```
       should return true for 40 x empty zero bytes
       should return true for 64 x empty zero bytes

LSP7
       should return true for 100 x empty zero bytes
    testing various non-zero bytes input
      when less than 32 bytes
  when using LSP7 contract with constructor
    when deploying the contract
         should return false with 4x random bytes
         should return false with 16x random bytes
      when abi-encoded array, with length = 0
         should return true with 64 bytes -> offset = 0x20, length = 0 (null)

LSP8
  when using LSP8 contract with constructor
    when deploying the contract
         should return true with 64 bytes -> offset = 0x20, length = 0 (null) + 10x extra zero bytes
         should return true with 64 bytes -> offset = 0x20, length = 0 (null) + 10x extra random bytes
      when abi-encoded array, with length = 1
         should return true with 1x array element - offset = 0x20, length = 1
         should return true with 1x array element - offset = 0x20, length = 1, +5 custom bytes in the end
    When Using Normal contracts
         should return true with 1x array element - offset = 0x25 (+ 5 custom bytes in between), length = 1
      when not correctly abi-encoded array, with length = 1
       should have deployed with the correct funding amount (undefined)
    when deploying the contract with or without value
       should calculate the address of a non-initializable contract correctly
         should return false with 1x array element - offset = 0x20, length = 1, but 31 bytes only
  AddressRegistry
       should calculate the address of an initializable contract correctly
         should return false with 1x array element - offset = 0x20, length = 1, but 30 bytes only
      when correctly abi-encoded array, but the length does not match the number of elements
     add address
       should calculate a different address of a contract if the salt changed
         should return false when 1x array element, but length = 2

       should calculate a different address of a contract if the initializeCalldata changed
и------------------------|--------------------------|--------------|-----------------------------и
|  Solc version: 0.8.10  и  Optimizer enabled: true  и  Runs: 1000  и  Block limit: 30000000 gas
ииииииииииииииииииииииии|ииииииииииииииииииииииииииии|ииииииииииииии|ииииииииииииииииииииииииииии
|  Methods
ииииииииииииии|иииииииииии|ииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии
|  Contract   и  Method  и  Min        и  Max        и  Avg         и  # calls    и  usd (avg)
и-------------|----------|-------------|-------------|--------------|-------------|--------------и

  19 passing (36s)

       should have deployed with the correct funding amount (0)
    when deploying the contract with or without value
       should calculate a different address of a contract if the bytecode changed
     add same address
       should revert when deploying a minimal proxy from `deployCreate2` function
       should have deployed with the correct funding amount (5)
    when deploying the contract
      when initializing the contract
       when the contract was initialized
          should support ERC165 interface
       should revert when deploying a CREATE2 contract with the same bytecode and salt
     should add and remove address
       should revert when sending value while deploying a non payable constructor contract
          should support ERC1271 interface
     should give the right count
          should support ERC725X interface
     get correct index
       should pass when sending value while deploying a payable constructor contract
          should support ERC725Y interface
     can list all values of the registry
       should revert when deploying a CREATE2 contract and passing calldata for a non-existing function where fallback function doesn't exist
     can get all raw values in one call
   AddressRegistryRequiresERC725
       should pass when deploying a CREATE2 contract and passing calldata for a non-existing function where fallback function exist
          should support LSP0 (ERC725Account) interface
          should have registered the ERC165 interface
          should support LSP1 interface
     add address
          should support ClaimOwnership interface
       should deploy an un-initializable CREATE2 contract and get the owner successfully
          should have set key 'SupportedStandards:LSP3UniversalProfile'
    when testing deployed contract
      when using `isValidSignature()` from ERC1271
       should revert when deploying with address(0) as owner
      once the contract was deployed
        when the contract was initialized
     external account adds address
       should revert when deploying with address(0) as owner
      once the contract was deployed
        when the contract was initialized

LSP6KeyManager
       should deploy an initializable CREATE2 contract and get the owner successfully
  when using LSP6KeyManager with constructor
    When Using Proxy contracts
    when deploying the contract
      when initializing the contract
       when the contract was initialized
       should calculate the address of a proxy correctly if it's initializable
     remove address
       should calculate the address of a proxy correctly if it's not initializable
         should verify signature from owner
       should calculate a different address of a proxy if the `salt` changed
       should calculate a different address of a proxy if its initializable or not
       should calculate a different address of a proxy if the `initializeCallData` changed
       should calculate a different address of a proxy if the `baseContract` changed
     should fail if called by a regular address

Test Custom implementation of ERC165Checker
       should revert when deploying a CREATE2 proxy contract with the same `baseContract` and salt
       should revert when sending value while deploying a CREATE2 proxy without `initializeCallData`
           should have registered the ERC165 interface
       should revert when deploying a proxy and sending value to a non payable function in deployCreate2Proxy
       should pass when deploying a proxy and sending value to a payable function in deployCreate2Proxy
         should fail when verifying signature from non-owner
       should revert when deploying a proxy and passing calldata for a non-existing function where fallback function doesn't exist
           should have registered the ERC165 interface
   Calling an EOA
   Calling an EOA
   Calling a contract without a fallback function that doesn't support ERC165
           should have registered the ERC725Y interface
           should have registered the ERC725X interface
   Calling a contract with a fallback function that doesn't support ERC165
       should pass when deploying a proxy and passing calldata for a non-existing function where fallback function exist
   Calling a contract that support ERC165 and ERC725X but doesn't support LSP1

Calculate LSP interfaces
       should deploy an un-initializable CREATE2 proxy contract and get the default owner successfully
           should have registered the ERC165 interface
   LSP0
   LSP1
   LSP1Delegate
           should have registered the LSP8 interface
   LSP6
   LSP7
   LSP8
   LSP9
   IClaimOwnership

Calculate ERC interfaces
           should have registered the ERC725Y interface
   ERC20
   ERC223
   ERC721
   ERC721Metadata
   ERC777
   ERC1155
   ERC1271

Key Manager gas cost interactions
  when using LSP6KeyManager with constructor
    after deploying the contract
         should return failValue when the owner doesn't support ERC1271
```

```
        when interacting with the ERC725Y storage
             should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when setting data on ERC725Y storage
         should deploy an initializable CREATE2 proxy contract and get the owner successfully

·-------------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                       Solc version: 0.8.10                        ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract          ·  Method                                        ·  Min       ·  Max         ·  Avg         ·  # calls      ·  usd (avg)
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalFactory  ·  deployCreate2(bytes,bytes32,bytes)             ·     70217  ·     2097638  ·      909058  ·             5  ·          -
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalFactory  ·  deployCreate2Proxy(address,bytes32,bytes)      ·     64473  ·      119156  ·       78256  ·             5  ·          -
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Deployments                                                         ·                                           ·  % of limit   ·
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP1UniversalReceiverDelegateUP                                     ·          -  ·           -  ·     1511074  ·          5 %  ·          -
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalFactory                                                    ·          -  ·           -  ·      680591  ·        2.3 %  ·          -
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile                                                    ·          -  ·           -  ·     2069065  ·        6.9 %  ·          -
····················|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfileInit                                                ·          -  ·           -  ·     2202316  ·        7.3 %  ·          -
·-------------------------------------------------------------------|------------|--------------|--------------|---------------|-------------·

  27 passing (36s)

Failed to generate 1 stack trace. Run Hardhat with --verbose to learn more.
            should have registered the LSP7 interface
           should revert when trying to edit Token Name
           should revert when trying to edit Token Symbol
        when minting tokens
          when tokenId has already been minted
             should have registered the ERC725Y interface
             should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when setting data on ERC725Y storage
           should set the 3 x keys for a basic UP setup => `LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`
           should revert when trying to edit Token Name
             should revert
          when tokenId has not been minted
            when `to` is the zero address
               should revert
            when `to` is not the zero address
           should revert when trying to edit Token Symbol
        when minting tokens
          when `to` is the zero address
gas cost LYX transfer - with 1 x allowed address:  74791
         when caller has only 1 x allowed address allowed
              should mint the token
        when tokens have been minted
          totalSupply
             should revert
          when `to` is not the zero address
gas cost LYX transfer - with 1 x allowed address + 1 x allowed standard:  85399
         when caller has only 1 x allowed address + 1 x allowed standard allowed
        display gas cost
             should have registered the LSP1Delegate interface
      when testing deployed contract
             should have registered the LSP9 interface
             should return total token supply
          balanceOf
            when the given address owns tokens
gas cost LYX transfer - everything allowed:  56030
           when caller has any allowed address and standard allowed

  NFTStorageMerkle
    Testing Merkle Tree
       Should return 8 for leaves count
       Keccak256 hash should match for the first NFT address
       Should verify the proof in the smart contract

             should mint the token amount
        when tokens have been minted
          totalSupply
·-------------------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                          Solc version: 0.8.10                           ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract                  ·  Method                                    ·  Min       ·  Max         ·  Avg         ·  # calls      ·  usd (avg)
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  AddressRegistry           ·  addAddress(address)                       ·     24140  ·      88613  ·       63942  ·             4  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  AddressRegistry           ·  removeAddress(address)                    ·          -  ·          -  ·       29976  ·             1  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  AddressRegistryRequiresERC725  ·  addAddress(address)                  ·          -  ·          -  ·       92208  ·             1  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP6KeyManager            ·  execute(bytes)                            ·     44069  ·      85399  ·       68073  ·             7  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile          ·  execute(uint256,address,uint256,bytes)    ·     35326  ·      99822  ·       67574  ·             2  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile          ·  setData(bytes32[],bytes[])                ·          -  ·          -  ·      448606  ·             1  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile          ·  transferOwnership(address)                ·          -  ·          -  ·       46163  ·             1  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Deployments                                                            ·                                           ·  % of limit   ·
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  AddressRegistry                                                        ·          -  ·          -  ·      394991  ·        1.3 %  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  AddressRegistryRequiresERC725                                          ·          -  ·          -  ·      514084  ·        1.7 %  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC725                                                                 ·          -  ·          -  ·     1444163  ·        4.8 %  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP6KeyManager                                                         ·          -  ·          -  ·     2446105  ·        8.2 %  ·          -
····························|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile                                                       ·          -  ·          -  ·     2091006  ·          7 %  ·          -
·-------------------------------------------------------------------------|-------------|-------------|--------------|---------------|-------------·

  36 passing (38s)

          should add +10 more LSP12IssuedAssets[]
              should return the owned token count
           when the given address does not own tokens
            should support ERC165 interface
            should return total token supply
          balanceOf
            when the given address owns tokens
               should return zero
          tokenOwnerOf
            when tokenId has not been minted
               should return the owned token count
            when the given address does not own tokens
             should support ERC1271 interface
             should have registered the LSP1 interface
               should revert
            when tokenId has been minted
               should return zero
          decimals
             should support LSP6 interface
           should add +1 LSP12IssuedAssets
              should return owner address
          tokenIdsOf
            when the given address owns some tokens
             should return 18 as default value
          authorizeOperator
            when operator is not the zero address
             should be linked to the right ERC725 account contract
      when testing deployed contract
        CHANGEOWNER
          when upgrading to a new KeyManager via transferOwnership(...)
            when caller does not have have CHANGEOWNER permission
               should return the list of owned tokenIds
            when the given address does not owns some tokens
        when testing ERC165 standard
           should support ERC165 interface
           should support LSP1Delegate interface
```

```
    when testing LSP7-DigitalAsset
           should succeed
          when operator is already authorized
           should return an empty list
      authorizeOperator
        when tokenId does not exist
         should support ClaimOwnership interface
       should add +1 LSP12IssuedAssets
      when minting tokens
        when minting 10 tokenA to universalProfile1
           should revert
        when caller is not owner of tokenId
             should succeed
        when operator is the zero address
           should revert
        when caller is owner of tokenId
          when operator is not the zero address
           should revert
    revokeOperator
      when operator is not the zero address
       should add +1 LSP12IssuedAssets
            should succeed
           when operator is already authorized
        should have set expected entries with ERC725Y.setData
  when testing deployed contract
           should revert
        when caller has ALL PERMISSIONS
         should succeed
      when operator is the zero address
               should revert
           when operator is the zero address
         should revert
      authorizedAmountFor
        when operator is the token owner
               should revert
    revokeOperator
      when tokenId does not exist
         should return the balance of the token owner
      when operator has not been authorized
     should add +1 LSP12IssuedAssets
         should revert
      when caller is not owner of tokenId
         should return zero
      when one account have been authorized
  when testing setting data
         should revert
      when caller is owner of tokenId
        when operator is not the zero address
         should return the authorized amount
      when many accounts have been authorized
     owner should be able to setData
         should register lsp5keys: arrayLength 1, index 0, tokenA address in UP1
      when minting 10 tokenB to universalProfile1
         should have set newKeyManager as pendingOwner
         should return the authorized amount for each operator
    transfers
      transfer
        when tokenOwner sends tx
          when using force=true
            when `to` is an EOA
              when `to` is not the zero address
     non-owner shouldn't be able to setData
     should add +1 LSP12IssuedAssets
           should succeed
        when operator is the zero address
           should revert
        when address provided to revoke is not an existing operator
           should revert
    isOperatorFor
      when tokenId has not been minted
     UniversalReceiverDelegate should be able to setData
  when using vault with UniversalProfile
    when transferring ownership of the vault to the universalProfile
                should allow transfering
             when `to` is the zero address
      should add +1 LSP12IssuedAssets
          should revert
       when tokenId has been minted
         when operator has not been authorized
            should return false
         when one account have been authorized for the tokenId
          owner should remain the current KeyManager
                  should revert
             when `to` is a contract
               when receiving contract supports LSP1
              should return true
         when many accounts have been authorized for the tokenId
      should add +1 LSP12IssuedAssets
          should register lsp5keys: arrayLength 2, index 1, tokenB address in UP1
       when minting 10 of the same tokenB to universalProfile1
            should return true for all operators
    getOperatorsOf
      when tokenId has not been minted
         should revert
      when tokenId has been minted
        when operator has not been authorized
         should keep the same lsp5keys: arrayLength 2, index 1, tokenB address in UP1
      when minting 10 tokenC to universalProfile1
           should return empty list
        when one account have been authorized for the tokenId
                  should allow transfering
               when receiving contract does not support LSP1
         should override the pendingOwner when transferOwnership(...) is called twice
        it should still be possible to call onlyOwner functions via the old KeyManager
           should return list
        when many accounts have been authorized for the tokenId
     should add +1 LSP12IssuedAssets
  when sending native tokens to the contract
       should register lsp10 keys of the vault on the profile
    when restricitng address to only talk to the vault
           should return list
    transfers
      transfer
        when tokenOwner sends tx
          when using force=true
            when `to` is an EOA
     should emit the right ValueReceived event
                should allow transfering
           when force=false
             when `to` is an EOA
     should allow to send a random payload as well, and emit the ValueReceived event
  when sending a random payload, without any value
              should revert
           when `to` is a contract
             when receiving contract supports LSP1
          setData(...)
     should execute the fallback function, but not emit the ValueReceived event
  when renouncing ownership of the universal profile
         should register lsp5keys: arrayLength 3, index 2, tokenC address in UP1
    when burning tokens
      when burning 10 tokenC (last token) from universalProfile1
       should allow friend to talk to the vault
                should allow transfering the tokenId
              when `to` is the zero address
        should fail when friend is interfacting with other contracts
    when renouncing ownership of the vault
                should allow transfering
              when receiving contract does not support LSP1
     should fail to confirm if delay didn't expire
           execute(...) - LYX transfer
      when caller has only CHANGE0OWNER permission
                should revert
             when `to` is a contract
               when receiving contract supports LSP1
                  should revert
           when the given amount is more than balance of tokenOwner
              should revert
         when operator sends tx
           when using force=true
             when `to` is an EOA
```

```
            when `to` is not the zero address
   should renounce ownership in a 2-step process after delay expires
when calling the `universalReceiver(...)` function
  from an EOA
     should fail to confirm if delay didn't expire
       should have set newKeyManager as pendingOwner
     should emit a UniversalReceiver(...) event with correct topics
  from a Contract
    via a contract call - `contract.universalReceiver(...)`
       should update lsp5keys: arrayLength 2, no map, no tokenC address in UP1
    when burning 10 tokenA (first token) from universalProfile1
               should allow transfering
             when `to` is the zero address
               should allow transfering the tokenId
             when receiving contract does not support LSP1
       should emit an UniversalReceiver(...) event
    via a low-level call - `address(contract).call(...)`
               should revert
           when `to` is a contract
             when receiving contract supports LSP1
       should emit an UniversalReceiver(...) event
when calling the `universalReceiver(...)` function while sending native tokens
  from an EOA
     should renounce ownership in a 2-step process after delay expires
when owner call transferOwnership(...)
       owner should remain the current KeyManager
   should have set the pendingOwner
     should emit a UniversalReceiver(...) event with correct topics
  from a Contract
    via a contract call - `contract.universalReceiver(...)`
               should allow transfering the tokenId
           when `from == to` address (= sending to tokenId's owner itself)
   owner should remain the current owner
               should allow transfering
             when receiving contract does not support LSP1
               should allow transfering
           when receiving contract does not support LSP1
       should emit an UniversalReceiver(...) event
    via a low-level call - `address(contract).call(...)`
   should override the pendingOwner when transferOwnership(...) is called twice
  it should still be allowed to call onlyOwner functions
            should revert
         when force=false
           when `to` is an EOA
       should override the pendingOwner when transferOwnership(...) is called twice
  when calling claimOwnership(...) from a KeyManager that is not the pendingOwner
       should pop and swap TokenA with TokenB, lsp5keys (tokenB should become first token) : arrayLength 1, index = 0, tokenB address in UP1
       should update lsp5keys: arrayLength 1, no map, no tokenA address in UP1
    when burning 10 (half of the amount) tokenB from universalProfile1
       should emit an UniversalReceiver(...) event
when owner call transferOwnership(...)
     setData(...)
               should allow transfering
         when force=false
           when `to` is an EOA
       should keep the same lsp5keys: arrayLength 1, index 0, tokenB address in UP1
    when burning 10 (remaining) tokenB from universalProfile1
     execute(...) - LYX transfer
when non-owner call transferOwnership(...)
             should not allow transfering the tokenId
           when `to` is a contract
             when receiving contract supports LSP1
   should revert
when calling claimOwnership(...)
   should have set the pendingOwner
   should revert when caller is not the pending owner
  when caller is the pending owner
             should revert
           when `to` is a contract
             when receiving contract supports LSP1
     should revert
  when calling claimOwnership(...) via the pending new KeyManager
   owner should remain the current owner
     should change the contract owner to the pendingOwner
   should override the pendingOwner when transferOwnership(...) is called twice
  it should still be allowed to call onlyOwner functions
     should have cleared the pendingOwner after transferring ownership
               should allow transfering the tokenId
             when receiving contract does not support LSP1
               should allow transfering
             when receiving contract does not support LSP1
     should have emitted a OwnershipTransferred event
  after pendingOwner has claimed ownership
    previous owner should not be allowed anymore to call onlyOwner functions
       should update lsp5keys: arrayLength 0, no map, no tokenB address in UP1
  when transferring tokens
     setData(...)
               should revert
          when the given amount is more than balance of tokenOwner
               should not allow transfering the tokenId
            when `from == to` address (= sending to tokenId's owner itself)
     should have change the account's owner to the pendingOwner (= pending KeyManager)
       should revert when calling `setData(...)`
           should revert
         when operator does not have enough authorized amount
     execute(...) - LYX transfer
when non-owner call transferOwnership(...)
            should revert
       when operator sends tx
         when using force=true
           when `to` is an EOA
         should revert when calling `execute(...)`
   should revert
when calling claimOwnership(...)
           should revert
       when the caller is not an operator
   should revert when caller is not the pending owner
  when caller is the pending owner
     should have cleared the pendingOwner after transfering ownership
  after KeyManager has been upgraded via claimOwnership(...)
    old KeyManager should not be allowed to call onlyOwner functions anymore
       should revert when calling `renounceOwnership(...)`
    new owner should be allowed to call onlyOwner functions
         should revert
    transferBatch
      when tokenOwner sends tx
        when using force=true
          when `to` is an EOA
            when `to` is the zero address
         should change the contract owner to the pendingOwner
          setData(...)
               should allow transfering the tokenId
             when `to` is the zero address
               should revert
             when `to` is not the zero address
          execute(...) - LYX transfer
when using LSP9Vault contract with proxy
  when deploying the base implementation contract
               should revert
             when `to` is a contract
               when receiving contract supports LSP1
         should have cleared the pendingOwner after transferring ownership
     prevent any address from calling the initialize(...) function on the implementation
  when deploying the contract as proxy
    when initializing the contract
      when the contract was initialized
           should revert when calling `setData(...)`
        should have emitted a OwnershipTransferred event
      after pendingOwner has claimed ownership
        previous owner should not be allowed anymore to call onlyOwner functions
               should allow transfering
             when `to` is a contract
               when receiving contract supports LSP1
         should have registered the ERC165 interface
           should revert when calling `setData(...)`
           should revert when calling `execute(...)`
        new Key Manager should be allowed to call onlyOwner functions
               should allow transfering the tokenId
             when receiving contract does not support LSP1
          should revert when calling `execute(...)`
               should allow transfering
             when receiving contract does not support LSP1
```

```
          should fund the universalProfle with 10 tokens (each) to test token transfers (TokenA, TokenB, TokenC)
          should register lsp5keys: arrayLength 3, index [1,2,3], [tokenA, tokenB, tokenC] addresses in UP1
        When transferring 10 (all) token A from UP1 to UP2
           should revert when calling `renounceOwnership(...)`
        new owner should be allowed to call onlyOwner functions
         should have registered the ERC725X interface
                   should allow transfering the tokenId
              when `from == to` address (= sending to tokenId's owner itself)
           setData(...)
           setData(...)
                   should allow transfering
            when force=false
              when `to` is an EOA
                 should revert
            when force=false
              when `to` is an EOA
                 should not allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
         should have registered the ERC725Y interface
                 should revert
              when `to` is a contract
                when receiving contract supports LSP1
           execute(...) - LYX transfer
when using UniversalProfile contract with proxy
  when deploying the base implementation contract
     prevent any address from calling the initialize(...) function on the implementation
  when deploying the proxy contract
    when initializing the proxy contract with or without value
           execute(...) - LYX transfer
    CHANGE / ADD permissions
      setting permissions keys (CHANGE vs ADD Permissions)
        when setting one permission key
          when caller is an address with ALL PERMISSIONS
       should have deployed with the correct funding amount (undefined)
      when the contract was initialized
         should support ERC165 interface
         should have registered the LSP9 interface
         should support ERC1271 interface
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
                   should allow transfering
                when receiving contract does not support LSP1
            should be allowed to ADD a permission
         should support ERC725X interface
                   should not allow transfering the tokenId
              when `from == to` address (= sending to tokenId's owner itself)
                   should revert
           when the given amount is more than balance of tokenOwner
         should support ERC725Y interface
         should have registered the LSP1 interface
            should be allowed to CHANGE a permission
         should support LSP0 (ERC725Account) interface
                 should revert
           when the caller is not an operator
              should revert
           when function parameters list length does not match
          should pop and swap TokenA with TokenC, lsp5keys (tokenC should become first token) : arrayLength 1, index = 0, tokenC address in UP1
          should update lsp5keys: arrayLength 2, no map, no tokenA address in UP1
          should register lsp5keys: arrayLength 1, index 0, tokenA address in UP2
       When transferring 5 (half of amount) token B from UP1 to UP2
         should support LSP1 interface
               should revert
          when the from address is incorrect
         should support ClaimOwnership interface
               should revert
          when operator sends tx
            when using force=true
              when `to` is an EOA
                when `to` is the zero address
              should be allowed to increment the 'AddressPermissions[]' key (length)
         should support ClaimOwnership interface
             should revert
          when the given tokenId has not been minted
         should have set key 'SupportedStandards:LSP3UniversalProfile'
when calling `initialize(...)` more than once
         should have set key 'SupportedStandards:LSP3UniversalProfile'
  when calling `initialize(...)` more than once
                 should revert
              when `to` is not the zero address
            should revert
        transferBatch
          when tokenOwner sends tx
            when using force=true
              when `to` is an EOA
              should be allowed to decrement the 'AddressPermissions[]' key (length)
          should have set expected entries with ERC725Y.setData
  when calling initialize more than once
     should revert
when deploying the proxy contract
  when initializing the proxy contract with or without value
                 should allow transfering
              when `to` is a contract
                when receiving contract supports LSP1
            should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP1
            should register lsp5keys: arrayLength 2, index 1, tokenB address in UP2
         When transferring 4 (few) token B from UP1 to UP2
              should be allowed to edit key at index -> AddressPermissions[4]
             if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
       should have deployed with the correct funding amount (0)
      when the contract was initialized
           should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP1
           should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP2
         When transferring 1 (remaining) token B from UP1 to UP2
          should support ERC165 interface
       should revert
when testing deployed contract
                  should allow transfering the tokenId
                when `to` is the zero address
          should support ERC1271 interface
                should revert
           when caller is an address with permission ADDPERMISSIONS
                   should allow transfering
                when receiving contract does not support LSP1
          should support ERC725X interface
                   should revert
              when `to` is a contract
                when receiving contract supports LSP1
             should support ERC725Y interface
      when testing setting data
              should be allowed to add a permission
           should support LSP0 (ERC725Account) interface
                   should allow transfering
              when force=false
                when `to` is an EOA
           should support LSP1 interface
                  should revert
              when `to` is a contract
                when receiving contract supports LSP1
            should update lsp5keys (no pop and swap as TokenB has the last index): arrayLength 1, no map, no tokenB address in UP1
           should support ClaimOwnership interface
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP2
          When transferring 10 (all) token C from UP1 to UP2
              should not be allowed to CHANGE a permission
           should have set key 'SupportedStandards:LSP3UniversalProfile'
      when calling `initialize(...)` more than once
         owner should be able to setData
         should revert
    when deploying the proxy contract
      when initializing the proxy contract with or without value
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
              should be allowed to increment the 'AddressPermissions[]' key (length)
                   should allow transfering
                when receiving contract does not support LSP1
         should have deployed with the correct funding amount (5)
        when the contract was initialized
           should support ERC165 interface
                   should revert
              when the given amount is more than balance of tokenOwner
           non-owner shouldn't be able to setData
                should not be allowed to decrement the 'AddressPermissions[]' key (length)
```

```
          should support ERC1271 interface
                should revert
             when function parameters list length does not match
          should support ERC725X interface
          should support ERC725Y interface
                    should allow transfering the tokenId
             when force=false
               when `to` is an EOA
              should not be allowed to edit key at index -> AddressPermissions[4]
             if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
                should revert
             when operator does not have enough authorized amount
          should support LSP0 (ERC725Account) interface
          should support LSP1 interface
                should revert
             when the caller is not an operator
        UniversalReceiverDelegate should be able to setData
            should update lsp5keys (no pop and swap as TokenC has the last index): arrayLength 0, no map, no tokenB address in UP1
      when using vault with UniversalProfile
        when transferring ownership of the vault to the universalProfile
            should register lsp5keys : arrayLength 3, index = 2, tokenC address in UP2
          When transferring 1 (few) token B from UP2 to UP1
                   should not allow transfering the tokenId
                when `to` is a contract
                  when receiving contract supports LSP1
                  should revert when trying to set a non-standard LSP6 permission data key
           when caller is an address with permission CHANGEPERMISSION
          should support ClaimOwnership interface
                should revert
    when using LSP7 contract with proxy
      when deploying the base implementation contract
         prevent any address from calling the initialize(...) function on the implementation
      when deploying the contract as proxy
            should have set key 'SupportedStandards:LSP3UniversalProfile'
        when calling `initialize(...)` more than once
         should revert when initializing with address(0) as owner
      when initializing the contract
        when the contract was initialized
         should revert
    when testing deployed contract
      when using `isValidSignature()` from ERC1271
               should not be allowed to ADD a permission
            should have registered the ERC165 interface
                should not be allowed to set (= ADD) a permission for an address that has 32 x 0 bytes (0x0000...0000) as permission value
                      should allow transfering the tokenId
                 when receiving contract does not support LSP1
         should verify signature from owner
             should register lsp5keys (UP1 able to re-register keys) : arrayLength 1, index = 0, tokenB address in UP1
        when removing all keys
           should have registered the ERC725Y interface
               should be allowed to CHANGE a permission
                     should not allow transfering the tokenId
              when the from address is incorrect
           should register lsp10 keys of the vault on the profile
        when restricitng address to only talk to the vault
         should fail when verifying signature from non-owner
                 should revert
              when the given tokenId has not been minted
               should not be allowed to increment the 'AddressPermissions[]' key (length)
           should have registered the LSP7 interface
                 should revert
              when function parameters list length does not match
               should be allowed to decrement the 'AddressPermissions[]' key (length)
           should have set expected entries with ERC725Y.setData
      when calling initialize more than once
         should return failValue when the owner doesn't support ERC1271
      when interacting with the ERC725Y storage
                should revert
             when operator sends tx
               when using force=true
                 when `to` is an EOA
           should allow friend to talk to the vault
         should revert
    when testing deployed contract
      when setting data on ERC725Y storage
               should be allowed to edit key at index -> AddressPermissions[4]
              if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
         should revert when trying to edit Token Name
                 should revert when trying to set a non-standard LSP6 permission data key
            when caller is an address with permission SETDATA
           should fail when friend is interfacting with other contracts
        when renouncing ownership of the vault
         should set the 3 x keys for a basic UP setup => `LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`
         should revert when trying to edit Token Symbol
      when minting tokens
        when `to` is the zero address
                 should allow transfering the tokenId
                when `to` is the zero address
               should not be allowed to ADD a permission
           should revert
        when `to` is not the zero address
                   should revert
                when `to` is a contract
                  when receiving contract supports LSP1
               should not be allowed to set (= ADD) a permission for an address that has 32 x 0 bytes (0x0000...0000) as permission value
           should mint the token amount
      when tokens have been minted
        totalSupply
               should not be allowed to CHANGE a permission
           should return total token supply
        balanceOf
          when the given address owns tokens
         should add +10 more LSP12IssuedAssets[]
               should not be allowed to increment the 'AddressPermissions[]' key (length)
              should return the owned token count
          when the given address does not own tokens
                   should allow transfering the tokenId
                  when receiving contract does not support LSP1
           should fail to confirm if delay didn't expire
               should not be allowed to decrement the 'AddressPermissions[]' key (length)
           should remove all lsp5 keys on both UP
      when testing LSP8-IdentifiableDigitalAsset
             should return zero
        decimals
        when minting tokens
          when minting tokenId 1 of tokenA to universalProfile1
           should return 18 as default value
        authorizeOperator
          when operator is not the zero address
               should not be allowed to edit key at index -> AddressPermissions[4]
              if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
         should add +1 LSP12IssuedAssets
                 should revert when trying to set a non-standard LSP6 permission data key
        setting Allowed Addresses
          when caller has permission ADDPERMISSIONS
             should succeed
            when operator is already authorized
                     should allow transfering the tokenId
              when force=false
                when `to` is an EOA
             should register lsp5keys: arrayLength 1, index 0, tokenA address in UP1
          when minting tokenId 1 of tokenB to universalProfile1
             should fail when trying to edit existing allowed addresses for an address
                  should not allow transfering the tokenId
                when `to` is a contract
                  when receiving contract supports LSP1
           should renounce ownership in a 2-step process after delay expires
    when owner call transferOwnership(...)
             should succeed
          when operator is the zero address
             should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of address[] initially
         should add +1 LSP12IssuedAssets
             should revert
    revokeOperator
      when operator is not the zero address
           should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed addresses
       should have set the pendingOwner
           should register lsp5keys: arrayLength 2, index 1, tokenB address in UP1
        when minting tokenId 2 of tokenB (another) to universalProfile1
           should keep the same lsp5keys: arrayLength 2, index 1, tokenB address in UP1
```

```
            when minting tokenId 1 of tokenC to universalProfile1
             should succeed
          when operator is the zero address
                      should allow transfering the tokenId
                   when receiving contract does not support LSP1
              should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed addresses
             should revert
          authorizedAmountFor
            when operator is the token owner
                      should not allow transfering the tokenId
                 when the from address is incorrect
           owner should remain the current owner
              should pass when beneficiary had no values set under AddressPermissions:AllowedAddresses:... + setting a valid abi-encoded array of address
[] (= with 12 x leading '00')
               when setting an invalid abi-encoded array of address[] for a new beneficiary
                should return the balance of the token owner
            when operator has not been authorized
                   should revert
                when the given tokenId has not been minted
               should register lsp5keys: arrayLength 3, index 2, tokenC address in UP1
          when burning tokens
            when burning tokenId 1 (all balance) of tokenC (last token) from universalProfile1
                 should revert with error when value = random bytes
               should return zero
            when one account have been authorized
                   should revert
                when function parameters list length does not match
           should add +1 LSP12IssuedAssets
                 should revert with error when value = invalid abi-encoded array of address[] (not enough leading zero bytes for an address -> 10 x '00')
            when caller has permission CHANGEPERMISSIONS
               should return the authorized amount
            when many accounts have been authorized
                   should revert
                when the caller is not an operator
           should override the pendingOwner when transferOwnership(...) is called twice
          it should still be allowed to call onlyOwner functions
              should fail when beneficiary had no values set under AddressPermissions:AllowedAddresses:...
              should update lsp5keys: arrayLength 2, no map, no tokenC address in UP1
            when burning tokenId 1 (all balance) of tokenA (first token) from universalProfile1
                   should revert
          _burn
            when tokenId has not been minted
               should return the authorized amount for each operator
          transfers
            transfer
              when tokenOwner sends tx
                when using force=true
                  when `to` is an EOA
                    when `to` is not the zero address
                should revert
            when tokenId has been minted
              after burning a tokenId
                 should have decreased the total supply
              should pass when trying to edit existing allowed addresses for an address
           should add +1 LSP12IssuedAssets
                 should have emitted a Transfer event with address(0) as `to` param
                when calling `tokenOwnerOf(...)` for the burnt tokenId
             setData(...)
                      should allow transfering
                   when `to` is the zero address
                   should revert stating tokenId does not exist
                when calling `tokenIdsOf(...)` with the initial owner address of the burnt token
              should pass when address had an invalid abi-encoded array of address[] initially
                  should return a list of tokenIds that does not contain the burnt tokenId
                when trying to get the operators for the burnt tokenId
              should pop and swap TokenA with TokenB, lsp5keys (tokenB should become first token) : arrayLength 1, index = 0, tokenB address in UP1
              should update lsp5keys: arrayLength 1, no map, no tokenA address in UP1
            when burning 1 tokenId (not all balance) of tokenB from universalProfile1
                      should revert
                   when `to` is a contract
                     when receiving contract supports LSP1
              should pass when address had 32 x 0 bytes set initially as allowed addresses
             execute(...) - LYX transfer
        when non-owner call transferOwnership(...)
                   should revert stating tokenId does not exist
    when using LSP8 contract with proxy
      when deploying the contract as proxy
         should revert when initializing with address(0) as owner
        when initializing the contract
          when the contract was initialized
               should keep the same lsp5keys: arrayLength 1, index 0, tokenB address in UP1
             when burning all tokenB from universalProfile1
            should add +1 LSP12IssuedAssets
              should have registered the ERC165 interface
              should have registered the ERC725Y interface
            should revert
        when calling claimOwnership(...)
                     should allow transfering
                   when receiving contract does not support LSP1
              should pass when address had 40 x 0 bytes set initially as allowed addresses
               when changing the list of allowed address of existing address to an invalid value
              should have registered the LSP8 interface
              should have set expected entries with ERC725Y.setData
        when calling initialize more than once
                 should revert with error when value = random bytes
               should update lsp5keys: arrayLength 0, no map, no tokenB address in UP1
          when transferring tokens
           should revert
      when testing deployed contract
        when setting data on ERC725Y storage
           should revert when caller is not the pending owner
          when caller is the pending owner
                      should allow transfering
                 when force=false
                   when `to` is an EOA
                  should revert with error when value = invalid abi-encoded array of address[] (not enough leading zero bytes for an address -> 10 x '00')
          setting Allowed Functions
            when caller has permission ADDPERMISSIONS
           should revert when trying to edit Token Name
           should add +1 LSP12IssuedAssets
           should revert when trying to edit Token Symbol
        when minting tokens
          when tokenId has already been minted
               should fail when trying to edit existing allowed functions for an address
                     should revert
                  when `to` is a contract
                    when receiving contract supports LSP1
             should change the contract owner to the pendingOwner
             should revert
          when tokenId has not been minted
            when `to` is the zero address
               should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of bytes4[] initially
               should revert
            when `to` is not the zero address
               should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed functions
                       should allow transfering
                    when receiving contract does not support LSP1
               should mint the token
        when tokens have been minted
          totalSupply
             should have cleared the pendingOwner after transferring ownership
               should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed functions
            should add +1 LSP12IssuedAssets
              should return total token supply
          balanceOf
            when the given address owns tokens
                      should revert
                 when the given amount is more than balance of tokenOwner
               should pass when beneficiary had no values set under AddressPermissions:AllowedFunctions:... + setting a valid abi-encoded array of bytes4[
] (= 28 leading zeros)
                when setting an invalid abi-encoded array of bytes4[] selector for a new beneficiary
              should return the owned token count
            when the given address does not own tokens
             should have emitted a OwnershipTransferred event
          after pendingOwner has claimed ownership
            previous owner should not be allowed anymore to call onlyOwner functions
                  should revert
              when operator sends tx
                when using force=true
                  when `to` is an EOA
```

```
                when `to` is not the zero address
           should return zero
     tokenOwnerOf
       when tokenId has not been minted
            should fail when setting an invalid abi-encoded array of bytes4[] (= random bytes)
        should fund the universalProfle with tokens to test token transfers (TokenA, TokenB, TokenC)
        should register lsp5keys: arrayLength 3, index [1,2,3], [tokenA, tokenB, tokenC] addresses in UP1
       When transferring tokenId 1 (all) of token A from UP1 to UP2
          should revert
       when tokenId has been minted
            should fail when setting an invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00')
       when caller has CHANGEPERMISSIONS
      should add +1 LSP12IssuedAssets
  when sending native tokens to the contract
       should revert when calling `setData(...)`
              should allow transfering
            when `to` is the zero address
       should return owner address
     tokenIdsOf
       when the given address owns some tokens
          should fail when beneficiary had no values set under AddressPermissions:AllowedFunctions:...
          should return the list of owned tokenIds
       when the given address does not owns some tokens
                should revert
           when `to` is a contract
             when receiving contract supports LSP1
      should emit the right ValueReceived event
          should pass when trying to edit existing allowed bytes4 selectors for an address
          should revert when calling `execute(...)`
          should return an empty list
     authorizeOperator
       when tokenId does not exist
          should pass when address had an invalid abi-encoded array of bytes4[] initially
          should revert
       when caller is not owner of tokenId
                should allow transfering
     should allow to send a random payload as well, and emit the ValueReceived event
  when sending a random payload, without any value
              when receiving contract does not support LSP1
          should pop and swap TokenA with TokenC, lsp5keys (tokenC should become first token) : arrayLength 1, index = 0, tokenC address in UP1
          should update lsp5keys: arrayLength 2, no map, no tokenA address in UP1
          should register lsp5keys: arrayLength 1, index 0, tokenA address in UP2
       When transferring tokenId 1 (not all balance) of token B from UP1 to UP2
          should revert when calling `renounceOwnership(...)`
       new owner should be allowed to call onlyOwner functions
          should revert
       when caller is owner of tokenId
         when operator is not the zero address
          should pass when address had 32 x 0 bytes set initially as allowed functions
     should execute the fallback function, but not emit the ValueReceived event
  when renouncing ownership of the universal profile
         should pass when address had 40 x 0 bytes set initially as allowed functions
         when changing the list of allowed bytes4 function selectors to an invalid value
                  should allow transfering
           when force=false
             when `to` is an EOA
            should succeed
           when operator is already authorized
          setData(...)
            should revert with error when value = random bytes
              should revert
           when operator is the zero address
                should revert
             when `to` is a contract
               when receiving contract supports LSP1
          should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP1
            should revert with error when value = invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00
')
       setting Allowed Standards
         when caller has ADDPERMISSIONS
            should register lsp5keys: arrayLength 2, index 1, tokenB address in UP2
         When transferring tokenId 2 (not all balance) of token B from UP1 to UP2
                should revert
       revokeOperator
         when tokenId does not exist
        should fail to confirm if delay didn't expire
            should fail when trying to edit existing allowed standards for an address
            should revert
         when caller is not owner of tokenId
            execute(...) - LYX transfer

·-------------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                      Solc version: 0.8.10                         ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Methods
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Contract          ·  Method      ·  Min  ·  Max  ·  Avg  ·  # calls  ·  usd (avg)
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC725            ·  execute(uint256,address,uint256,bytes)  ·  -  ·  -  ·  40230  ·  3  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC725            ·  renounceOwnership()  ·  31844  ·  49715  ·  43758  ·  6  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC725            ·  setData(bytes32,bytes)  ·  51375  ·  51603  ·  51438  ·  4  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  ERC725            ·  transferOwnership(address)  ·  31713  ·  48813  ·  46801  ·  17  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP0ERC725Account ·  claimOwnership()  ·  -  ·  -  ·  35569  ·  9  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP0ERC725AccountInit  ·  initialize(address)  ·  -  ·  -  ·  75256  ·  35  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP6KeyManager    ·  execute(bytes)  ·  44069  ·  204003  ·  57913  ·  50  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault         ·  claimOwnership()  ·  -  ·  -  ·  32865  ·  9  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault         ·  execute(uint256,address,uint256,bytes)  ·  -  ·  -  ·  37550  ·  3  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault         ·  renounceOwnership()  ·  27176  ·  47071  ·  40439  ·  6  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault         ·  setData(bytes32,bytes)  ·  48641  ·  48857  ·  48701  ·  4  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault         ·  transferOwnership(address)  ·  29029  ·  46141  ·  44129  ·  17  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile  ·  setData(bytes32[],bytes[])  ·  244284  ·  244320  ·  244318  ·  44  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile  ·  transferOwnership(address)  ·  46139  ·  46163  ·  46162  ·  44  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  Deployments                                                  ·                                 ·  % of limit   ·
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP1UniversalReceiverDelegateUP                              ·  -  ·  -  ·  1511074  ·  5 %  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP6KeyManager                                               ·  2446093  ·  2446105  ·  2446104  ·  8.2 %  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9Vault                                                    ·  -  ·  -  ·  2000952  ·  6.7 %  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  LSP9VaultInit                                                ·  -  ·  -  ·  2097304  ·  7 %  ·  -
····················|··············|·····|·····|·····|··|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|·····|
|  UniversalProfile                                             ·  -  ·  -  ·  2091006  ·  7 %  ·  -
·-------------------------------------------------------------------|-------------|-------------|--------------|---------------|-------------·

  62 passing (2m)

                    should allow transfering
                when receiving contract does not support LSP1
            should revert
          when caller is owner of tokenId
            when operator is not the zero address
             should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of bytes4[] initially
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP1
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP2
          When transferring tokenId 3 (remaining balance) of token B from UP1 to UP2
        should renounce ownership in a 2-step process after delay expires
      when calling the `universalReceiver(...)` function
        from an EOA
                  should revert
               when the given amount is more than balance of tokenOwner
             should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed functions
               should succeed
             when operator is the zero address
              should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed functions
                should revert
```

```
              when address provided to revoke is not an existing operator
                   should revert
                 when operator does not have enough authorized amount
               should emit a UniversalReceiver(...) event with correct topics
          from a Contract
            via a contract call - `contract.universalReceiver(...)`
                 should revert
          isOperatorFor
            when tokenId has not been minted
               should update lsp5keys (no pop and swap as TokenB has the last index): arrayLength 1, no map, no tokenB address in UP1
               should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in UP2
            When transferring tokenId 1 (all balance) of token C from UP1 to UP2
               should pass when beneficiary had no values set under AddressPermissions:AllowedStandards:... + setting a valid abi-encoded array of bytes4[
] (= 28 leading zeros)
              when setting an invalid abi-encoded array of bytes4[] interface IDs for a new beneficiary
                   should revert
              when the caller is not an operator
               should revert
            when tokenId has been minted
              when operator has not been authorized
               should emit an UniversalReceiver(...) event
            via a low-level call - `address(contract).call(...)`
                 should fail when setting an invalid abi-encoded array of bytes4[] (= random bytes)
                 should return false
              when one account have been authorized for the tokenId
                 should revert
            transferBatch
              when tokenOwner sends tx
                when using force=true
                  when `to` is an EOA
                    when `to` is the zero address
                 should fail when setting an invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00')
            when caller has CHANGEPERMISSION
                 should return true
              when many accounts have been authorized for the tokenId
                 should return true
               when many accounts have been authorized for the tokenId
                should emit an UniversalReceiver(...) event
        when calling the `universalReceiver(...)` function while sending native tokens
          from an EOA
               should fail when beneficiary had no values set under AddressPermissions:AllowedStandards:...
                       should revert
                    when `to` is not the zero address
                  should return true for all operators
          getOperatorsOf
            when tokenId has not been minted
               should update lsp5keys (no pop and swap as TokenC has the last index): arrayLength 0, no map, no tokenB address in UP1
               should register lsp5keys : arrayLength 3, index = 2, tokenC address in UP2
            When transferring 1 tokenId (not all balance) of token B from UP2 to UP1
               should pass when trying to edit existing allowed standards for an address
             should emit a UniversalReceiver(...) event with correct topics
          from a Contract
            via a contract call - `contract.universalReceiver(...)`
               should revert
            when tokenId has been minted
              when operator has not been authorized
               should pass when address had an invalid abi-encoded array of bytes4[] interface IDs initially
                       should allow transfering
                  when `to` is a contract
                    when receiving contract supports LSP1
                  should return empty list
               when one account have been authorized for the tokenId
                should emit an UniversalReceiver(...) event
            via a low-level call - `address(contract).call(...)`
               should pass when address had 32 x 0 bytes set initially as allowed standards
                 should return list
               when many accounts have been authorized for the tokenId
                should register lsp5keys (UP1 able to re-register keys) : arrayLength 1, index = 0, tokenB address in UP1
        when testing LSP9-Vault
               should pass when address had 40 x 0 bytes set initially as allowed standards
              when changing the list of allowed bytes4 interface IDs to an invalid value
          when transferring ownership of vaults from EOA to UP
            When transfering Ownership of VaultA to UP1
                       should allow transfering
                    when receiving contract does not support LSP1
              should emit an UniversalReceiver(...) event
        when owner call transferOwnership(...)
                 should revert with error when value = random bytes
        when owner call transferOwnership(...)
                 should revert with error when value = random bytes
                 should return list
          transfers
            transfer
              when tokenOwner sends tx
                when using force=true
                  when `to` is an EOA
                  should revert with error when value = invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00
')
          setting Allowed ERC725YKeys
            when caller has ADDPERMISSIONS
              when beneficiary had some ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
           should have set the pendingOwner
               should register lsp10key: arrayLength 1, index 0, VaultA address in UP1
            When transfering Ownership of VaultB to UP1
                 should fail when adding an extra allowed ERC725Y data key
                       should allow transfering
                when force=false
                  when `to` is an EOA
                     should allow transfering the tokenId
                    when `to` is the zero address
                 should fail when removing an allowed ERC725Y data key
           owner should remain the current owner
                    should revert
                 when `to` is a contract
                   when receiving contract supports LSP1
                 should fail when trying to clear the array completely
              should register lsp10key: arrayLength 1, index 0, VaultA address in UP1
            When transfering Ownership of VaultC to UP1
                       should revert
                 when `to` is a contract
                   when receiving contract supports LSP1
                 should fail when setting an invalid abi-encoded array of bytes32[]
             when beneficiary had no ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
           should override the pendingOwner when transferOwnership(...) is called twice
          it should still be allowed to call onlyOwner functions
                 should pass when setting a valid abi-encoded array of bytes32[]
              should register lsp10key: arrayLength 1, index 0, VaultA address in UP1
          when transferring ownership of vaults from UP to UP
            When transfering Ownership of VaultA from UP1 to UP2
                       should allow transfering
                    when receiving contract does not support LSP1
                 should fail when setting an invalid abi-encoded array of bytes32[] (random bytes)
             when caller has CHANGEPERMISSIONS
               when beneficiary had some ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
                       should allow transfering the tokenId
                    when receiving contract does not support LSP1
             setData(...)
                 should pass when adding an extra allowed ERC725Y data key
                       should revert
                  when the given amount is more than balance of tokenOwner
                       should allow transfering the tokenId
                   when `from == to` address (= sending to tokenId's owner itself)
                 should pass when removing an allowed ERC725Y data key
             execute(...) - LYX transfer
        when non-owner call transferOwnership(...)
                  should revert
                when function parameters list length does not match
                     should revert
                when force=false
                  when `to` is an EOA
                  should pass when trying to clear the array completely
           should revert
        when calling claimOwnership(...)
                  should revert
              when operator sends tx
                when using force=true
                  when `to` is an EOA
                    when `to` is the zero address
                 should fail when setting an invalid abi-encoded array of bytes32[]
              when beneficiary had no ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
               should pop and swap VaultA with VaultC, lsp10keys (VaultC should become first vault) : arrayLength 2, index = 0, VaultC address in UP1
               should register lsp10key: arrayLength 1, index 0, VaultA address in UP2
             When transfering Ownership of VaultB from UP1 to UP2
                     should not allow transfering the tokenId
```

```
        when `to` is a contract
          when receiving contract supports LSP1
 should revert when caller is not the pending owner
when caller is the pending owner
       should fail and not authorize to add a list of allowed ERC725Y data keys (not authorised)
            should revert
          when `to` is not the zero address
       should fail when setting an invalid abi-encoded array of bytes32[]
setting mixed keys (SETDATA, CHANGE & ADD Permissions
  when setting multiple keys
    when caller is an address with ALL PERMISSIONS
   should change the contract owner to the pendingOwner
            should allow transfering the tokenId
          when receiving contract does not support LSP1
            should allow transfering
        when `to` is a contract
          when receiving contract supports LSP1
       (should pass): 2 x keys + add 2 x new permissions
     should update lsp10keys (no pop and swap as VaultB has the last index): arrayLength 1, no map, no VaultB address in UP1
     should register lsp10key: arrayLength 2, index 1, VaultB address in UP2
  When transfering Ownership of VaultC from UP1 to UP2
            should not allow transfering the tokenId
         when `from == to` address (= sending to tokenId's owner itself)
   should have cleared the pendingOwner after transferring ownership
       (should pass): 2 x keys + change 2 x existing permissions
           should revert
    when operator sends tx
      when using force=true
        when `to` is an EOA
            should allow transfering
          when receiving contract does not support LSP1
       (should pass): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
    when caller is an address with permission SETDATA + ADDPERMISSIONS
   should have emitted a OwnershipTransferred event
after pendingOwner has claimed ownership
  previous owner should not be allowed anymore to call onlyOwner functions
     should remove all lsp10keys : arrayLength 0, no map, no VaultC address in UP1
     should register lsp10key: arrayLength 3, index 2, VaultC address in UP2
  When transferring Ownership of VaultB from UP2 to UP1
          should allow transfering the tokenId
         when `to` is the zero address
            should allow transfering
      when force=false
        when `to` is an EOA
       (should pass): 2 x keys + add 2 x new permissions + increment AddressPermissions[].length by +2
     should revert when calling `setData(...)`
            should revert
        when `to` is a contract
          when receiving contract supports LSP1
       (should fail): 2 x keys + add 2 x new permissions + decrement AddressPermissions[].length by -1
           should revert
        when `to` is a contract
          when receiving contract supports LSP1
       (should fail): 2 x keys + change 2 x existing permissions
     should revert when calling `execute(...)`
       (should fail): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
    when caller is an address with permission SETDATA + CHANGEPERMISSIONS
            should allow transfering the tokenId
          when receiving contract does not support LSP1
            should allow transfering
          when receiving contract does not support LSP1
     should register lsp10key (UP1 able to re-write) : arrayLength 1, index 0, VaultB address in UP1
when transferring ownership of vaults from UP to EOA
  When transfering Ownership of VaultA from UP2 to EOA
     should revert when calling `renounceOwnership(...)`
  new owner should be allowed to call onlyOwner functions
       (should pass): 2 x keys + remove 2 x addresses with permissions + decrement AddressPermissions[].length by -2
            should revert
      when the given amount is more than balance of tokenOwner
     setData(...)
     should pop and swap VaultA with VaultC, lsp10keys (VaultC should become first vault) : arrayLength 1, index = 0, VaultC address in UP2
when deploying vault to a UP directly
       (should pass): 2 x keys + change 2 x existing permissions
            should allow transfering the tokenId
        when `from == to` address (= sending to tokenId's owner itself)
   should register the data key relevant to the vault deployed in the UP storage
                  when `from == to` address (= sending to tokenId's owner itself)
            should register the data key relevant to the vault deployed in the UP storage

  LSP1UniversalReceiverDelegateVault
    when testing deployed contract
                 should revert
               when function parameters list length does not match
                (should fail): 2 x keys + add 2 x new permissions
      when testing ERC165 standard
         should support ERC165 interface
         should support LSP1Delegate interface
      when testing LSP7-DigitalAsset
        when minting tokens
          when minting 10 tokenA to lsp9Vault1
                {should fail): 2 x keys + increment AddressPermissions[].length by +1
                    should revert
               when force=false
                 when `to` is an EOA
              execute(...) - LYX transfer

·-----------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                       Solc version: 0.8.10                      ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  Methods
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  Contract               ·  Method                                 ·  Min        ·  Max        ·  Avg         ·  # calls      ·  usd (avg)
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  ERC725                 ·  execute(uint256,address,uint256,bytes)  ·          -  ·          -  ·       40185  ·            3  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  ERC725                 ·  renounceOwnership()                     ·      31844  ·      49715  ·       43758  ·            6  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  ERC725                 ·  setData(bytes32,bytes)                  ·          -  ·          -  ·       51378  ·            2  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  ERC725                 ·  setData(bytes32[],bytes[])              ·     245245  ·     584940  ·      459221  ·           10  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  ERC725                 ·  transferOwnership(address)              ·      31713  ·      48813  ·       46801  ·           17  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  LSP0ERC725Account      ·  claimOwnership()                        ·          -  ·          -  ·       31238  ·            9  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  LSP0ERC725Account      ·  universalReceiver(bytes32,bytes)        ·      31070  ·      31526  ·       31298  ·            4  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  LSP0ERC725AccountInit  ·  initialize(address)                     ·          -  ·          -  ·       74535  ·          149  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  claimOwnership()                        ·          -  ·          -  ·       28545  ·            9  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  execute(uint256,address,uint256,bytes)  ·          -  ·          -  ·       37505  ·            3  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  renounceOwnership()                     ·      27194  ·      47093  ·       40460  ·            6  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  setData(bytes32,bytes)                  ·          -  ·          -  ·       48665  ·            2  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  setData(bytes32[],bytes[])              ·     242444  ·     581777  ·      456178  ·           10  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  transferOwnership(address)              ·      29063  ·      46163  ·       44151  ·           17  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile       ·  universalReceiver(bytes32,bytes)        ·      28405  ·      28855  ·       28630  ·            4  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  Deployments                                                       ·                                          ·  % of limit   ·
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfile                                                  ·          -  ·          -  ·     2091006  ·          7 %  ·          -
··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········|··········
|  UniversalProfileInit                                              ·          -  ·          -  ·     2202316  ·        7.3 %  ·          -
·-----------------------------------------------------------------|-------------|-------------|--------------|---------------|-------------·

  120 passing (3m)

                  should revert
               when operator does not have enough authorized amount
                (should fail): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
       SETDATA
         when caller is an EOA
           when setting one key
             For UP owner
            should register lsp5keys: arrayLength 1, index 0, tokenA address in Vault1
         when minting 10 tokenB to lsp9Vault1
                   should not allow transfering the tokenId
```

```
                when `to` is a contract
                  when receiving contract supports LSP1
             should register lsp5keys: arrayLength 2, index 1, tokenB address in Vault1
          when minting 10 of the same tokenB to lsp9Vault1
                should pass
              For address that has permission SETDATA
             should keep the same lsp5keys: arrayLength 2, index 1, tokenB address in Vault1
          when minting 10 tokenC to lsp9Vault1
                 should revert
               when the caller is not an operator
             should register lsp5keys: arrayLength 3, index 2, tokenC address in Vault1
        when burning tokens
          when burning 10 tokenC (last token) from lsp9Vault1
                should pass
              For address that doesn't have permission SETDATA
                 should not allow
            when setting multiple keys
              For UP owner
             should update lsp5keys: arrayLength 2, no map, no tokenC address in Vault1
          when burning 10 tokenA (first token) from lsp9Vault1
                 should revert

LSP7CappedSupply
  when using LSP7CappedSupply with constructor
    when deploying the contract
      when the contract was initialized
         should have registered the ERC165 interface
                     should allow transfering the tokenId
                  when receiving contract does not support LSP1
         should have registered the ERC725Y interface
         should have registered the LSP7 interface
               (should pass): adding 5 singleton keys
         should have set expected entries with ERC725Y.setData
    when testing deployed contract
      tokenSupplyCap
         should allow reading tokenSupplyCap
      when minting tokens
         should allow minting amount up to tokenSupplyCap
        when cap has been reached
          should error when minting more than tokenSupplyCapTokens
          should pop and swap TokenA with TokenB, lsp5keys (tokenB should become first token) : arrayLength 1, index = 0, tokenB address in Vault1
                    should not allow transfering the tokenId
               when `from == to` address (= sending to tokenId's owner itself)
          should update lsp5keys: arrayLength 1, no map, no tokenA address in Vault1
        when burning 10 (half of the amount) tokenB from lsp9Vault1
          should keep the same lsp5keys: arrayLength 1, index 0, tokenB address in Vault1
        when burning 10 (remaining) tokenB from lsp9Vault1
                  should revert
             when the caller is not an operator
          should allow minting after burning
when using LSP7CappedSupply with proxy
  when deploying the contract as proxy
    when initializing the contract
      when the contract was initialized
         should update lsp5keys: arrayLength 0, no map, no tokenB address in Vault1
    when transferring tokens
           (should pass): adding 10 LSP12IssuedAssets
         should have registered the ERC165 interface
         should have registered the ERC725Y interface
               should revert
          when the from address is incorrect
         should have registered the LSP7 interface
         should have set expected entries with ERC725Y.setData
    when calling initialize more than once
           should revert
          when the given tokenId has not been minted
       should revert
  when testing deployed contract
    tokenSupplyCap
           (should pass): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
         For address that has permission SETDATA
       should allow reading tokenSupplyCap
    when minting tokens
           should revert
        transferBatch
          when tokenOwner sends tx
            when using force=true
              when `to` is an EOA
         should fund the universalProfle with 10 tokens (each) to test token transfers (TokenA, TokenB, TokenC)
         should register lsp5keys: arrayLength 3, index [1,2,3], [tokenA, tokenB, tokenC] addresses in Vault1
        When transferring 10 (all) token A from UP1 to UP2
         should allow minting amount up to tokenSupplyCap
        when cap has been reached
               (should pass): adding 5 singleton keys
          should error when minting more than tokenSupplyCapTokens
          should allow minting after burning

  LSP7CompatibleERC20
    when using LSP7 contract with constructor
      when deploying the contract
        when initializing the contract
          when the contract was initialized
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should have registered its ERC165 interface
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should have set expected entries with ERC725Y.setData
      when testing deployed contract
        approve
          when operator is the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should revert
          when the operator had no authorized amount
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should pop and swap TokenA with TokenC, lsp5keys (tokenC should become first token) : arrayLength 1, index = 0, tokenC address in Vault1
             should update lsp5keys: arrayLength 2, no map, no tokenA address in Vault1
             should succeed by setting the given amount
          when the operator had an authorized amount
            when the operator authorized amount is changed to another non-zero value
             should register lsp5keys: arrayLength 1, index 0, tokenA address in Vault2
          When transferring 5 (half of amount) token B from UP1 to UP2
                (should pass): adding 10 LSP12IssuedAssets
                    should allow transfering the tokenId
                   when `to` is the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
               should succeed by replacing the existing amount with the given amount
            when the operator authorized amount is changed to zero
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault1
             should register lsp5keys: arrayLength 2, index 1, tokenB address in Vault2
          When transferring 4 (few) token B from UP1 to UP2
               should succeed by replacing the existing amount with the given amount
        allowance
          when operator has been approved
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault1
             should return approval amount
          when operator has not been approved
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault2
          When transferring 1 (remaining) token B from UP1 to UP2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should return zero
        mint
          when a token is minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should have expected events
        burn
          when a token is burned
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                (should pass): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
              For address that doesn't have permission SETDATA
             should have expected events
        transfers
          transfer
            when sender has enough balance
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                     should revert
                when `to` is a contract
```

```
                  when receiving contract supports LSP1
                should allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
                 (should fail): adding 5 singleton keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should update lsp5keys (no pop and swap as TokenB has the last index): arrayLength 1, no map, no tokenB address in Vault1
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault2
          When transferring 10 (all) token C from UP1 to UP2
                  should allow transfering the tokenId
               when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                 (should fail): adding 10 LSP12IssuedAssets
                  should allow transfering the tokenId
            when sender does not have enough balance
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
              should revert
          transferFrom
            when sender has enough balance
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                 (should fail): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
          when caller is a contract
            > contract calls
                  should allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should update lsp5keys (no pop and swap as TokenC has the last index): arrayLength 0, no map, no tokenB address in Vault1
             should register lsp5keys : arrayLength 3, index = 2, tokenC address in Vault2
          When transferring 1 (few) token B from UP2 to UP1
               should allow to set a key hardcoded inside a function of the calling contract
                      should allow transfering the tokenId
                   when receiving contract does not support LSP1
                  should allow transfering the tokenId
                when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                Should allow to set a key computed inside a function of the calling contract
                    should allow transfering the tokenId
              when sender does not have enough balance
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                should revert
    when using LSP7 contract with proxy
      when deploying the base implementation contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
         prevent any address from calling the initialize(...) function on the implementation
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should register lsp5keys (UP1 able to re-register keys) : arrayLength 1, index = 0, tokenB address in Vault1
        when removing all keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                Should allow to set a key computed from parameters given to a function of the calling contract
            > Low-level calls
              should have registered its ERC165 interface
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should have set expected entries with ERC725Y.setData
        when calling initialize more than once
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                Should allow to `setHardcodedKeyRawCall` on UP
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
           should revert
      when testing deployed contract
        approve
          when operator is the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                       should allow transfering the tokenId
                 when force=false
                   when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                Should allow to `setComputedKeyRawCall` on UP
             should revert
          when the operator had no authorized amount
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should succeed by setting the given amount
          when the operator had an authorized amount
            when the operator authorized amount is changed to another non-zero value
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                Should allow to `setComputedKeyFromParamsRawCall` on UP
          when caller is another UniversalProfile (with a KeyManager attached as owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                should succeed by replacing the existing amount with the given amount
            when the operator authorized amount is changed to zero
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                     should not allow transfering the tokenId
                  when `to` is a contract
                    when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             Alice should have ALL PERMISSIONS in her UP
               should succeed by replacing the existing amount with the given amount
        allowance
          when operator has been approved
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should return approval amount
          when operator has not been approved
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
           should remove all lsp5 keys on both UP
      when testing LSP8-IdentifiableDigitalAsset
             Bob should have ALL PERMISSIONS in his UP
             should return zero
        mint
          when a token is minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
        when minting tokens
          when minting tokenId 1 of tokenA to lsp9Vault1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                      should allow transfering the tokenId
                    when receiving contract does not support LSP1
             should have expected events
        burn
          when a token is burned
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             Alice's UP should have permission SETDATA on Bob's UP
             should have expected events
        transfers
          transfer
            when sender has enough balance
              when `to` is an EOA
             should register lsp5keys: arrayLength 1, index 0, tokenA address in Vault1
          when minting tokenId 1 of tokenB to lsp9Vault1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                      should not allow transfering the tokenId
                when the from address is incorrect
                  should allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
             should register lsp5keys: arrayLength 2, index 1, tokenB address in Vault1
          when minting tokenId 2 of tokenB (another) to lsp9Vault1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
gas used:  97527
             Alice's UP should be able to `setData(...)` on Bob's UP
           when caller has SUPER_SETDATA + some allowed ERC725YKeys
             when trying to set a disallowed key
              should keep the same lsp5keys: arrayLength 2, index 1, tokenB address in Vault1
          when minting tokenId 1 of tokenC to lsp9Vault1
               should be allowed to set a disallowed key: 0xce4b625ea0e12f6f6c6897ae6135917a8a1ab2f3d3e8bb1b799d3629495d4a52
                   should revert
                when the given tokenId has not been minted
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should register lsp5keys: arrayLength 3, index 2, tokenC address in Vault1
        when burning tokens
          when burning tokenId 1 (all balance) of tokenC (last token) from lsp9Vault1
               should be allowed to set a disallowed key: 0x30520d24ed35d205e2401d763abf84056b914487855139e6b5a38a3eb53a5f72
               should be allowed to set a disallowed key: 0xd725eaad24b642ab5ec92f479215e2c326d0fb18fc80ac1413070b4e913ea897
                   should allow transfering the tokenId
            when sender does not have enough balance
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
             should update lsp5keys: arrayLength 2, no map, no tokenC address in Vault1
          when burning tokenId 1 (all balance) of tokenA (first token) from lsp9Vault1
                   should revert
                when function parameters list length does not match
               should be allowed to set a disallowed key: 0xded639d04e7d7a12082e6ceae9b28efca1539ebc0cd3f630e0d039281df45fc3
               should revert
          transferFrom
            when sender has enough balance
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
               should be allowed to set a disallowed key: 0x9f0ee8fd4a734ff11929291ce258b12e5c2fc09653010ebac39dbf114ee5de73
            when trying to set an allowed key
             should pop and swap TokenA with TokenB, lsp5keys (tokenB should become first token) : arrayLength 1, index = 0, tokenB address in Vault1
             should update lsp5keys: arrayLength 1, no map, no tokenA address in Vault1
          when burning 1 tokenId (not all balance) of tokenB from lsp9Vault1
                 should allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
               should be allowed to set the 1st allowed key
                   should revert
              when operator sends tx
                when using force=true
                  when `to` is an EOA
             should keep the same lsp5keys: arrayLength 1, index 0, tokenB address in Vault1
          when burning all tokenB from lsp9Vault1
               should be allowed to set the 2nd allowed key
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
               should be allowed to set the 3rd allowed key
        CALL
          when interacting via `execute(...)`
            when caller has ALL PERMISSIONS
             should update lsp5keys: arrayLength 0, no map, no tokenB address in Vault1
        when transferring tokens
                   should allow transfering the tokenId
            when sender does not have enough balance
               should pass and change state at the target contract
            when caller has permission CALL
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,uint256,bool,bytes))
                     should allow transfering the tokenId
                    when `to` is the zero address
               should pass and change state at the target contract
            when caller does not have permission CALL
               should revert

  LSP7Mintable
    when using LSP7Mintable with constructor
      when deploying the contract
        when the contract was initialized
           should have registered the ERC165 interface
           should have registered the ERC725Y interface
           should have registered the LSP7 interface
               should revert
            when calling a function that returns some value
           should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when owner minting tokens
           should increase the total supply
           should increase the tokenReceiver balance
        when non-owner minting tokens
           should revert
        when owner try to re-enter mint function through the UniversalReceiverDelegate
               should return the value to the Key Manager <- UP <- targetContract.getName()
                       should revert
                  when `to` is a contract
                    when receiving contract supports LSP1
               Should return the value to the Key Manager <- UP <- targetContract.getNumber()
            when calling a function that reverts
           should pass
    when using LSP7Mintable with proxy
      when deploying the base implementation contract
           should fund the universalProfle with tokens to test token transfers (TokenA, TokenB, TokenC)
           should register lsp5keys: arrayLength 3, index [1,2,3], [tokenA, tokenB, tokenC] addresses in Vault1
          When transferring tokenId 1 (all) of token A from UP1 to UP2
         prevent any address from calling the initialize(...) function on the implementation
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
               should revert
          when interacting via `executeRelayCall(...)`
            when signer has ALL PERMISSIONS
             should have registered the ERC165 interface
             should have registered the ERC725Y interface
               should execute successfully
            when signer has permission CALL
             should have registered the LSP7 interface
               should execute successfully
            when signer does not have permission CALL
             should have set expected entries with ERC725Y.setData
        when calling initialize more than once
                       should allow transfering the tokenId
                    when receiving contract does not support LSP1
           should revert
      when testing deployed contract
               should fail
        STATICCALL
          when caller has ALL PERMISSIONS
             should pop and swap TokenA with TokenC, lsp5keys (tokenC should become first token) : arrayLength 1, index = 0, tokenC address in Vault1
             should update lsp5keys: arrayLength 2, no map, no tokenA address in Vault1
             should register lsp5keys: arrayLength 1, index 0, tokenA address in Vault2
          When transferring tokenId 1 (not all balance) of token B from UP1 to UP2
        when owner minting tokens
           should increase the total supply
           should increase the tokenReceiver balance
        when non-owner minting tokens
           should revert
        when owner try to re-enter mint function through the UniversalReceiverDelegate
             should pass and return data
          when caller has permission STATICCALL
             should pass and return data
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault1
             should register lsp5keys: arrayLength 2, index 1, tokenB address in Vault2
          When transferring tokenId 2 (not all balance) of token B from UP1 to UP2
                       should allow transfering the tokenId
                when force=false
                  when `to` is an EOA
             should revert when trying to change state at the target contract
             should revert when caller try to make a CALL
          when caller does not have permission STATICCALL
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault1
           should pass
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault2
          When transferring tokenId 3 (remaining balance) of token B from UP1 to UP2

и-------------------------------------------------------------------------------------------------|---------------------------|--------------|-------------
----------------и
|                                      Solc version: 0.8.10                                       и  Optimizer enabled: true  и  Runs: 1000  и  Block limit
: 30000000 gas
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииииииииииииииииии|ииииииииииииии|иииииииииииии
ииииииииииииииииии
|  Methods

ииииииииииииииииииииииииииииииииииииии|ииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|ииииииииииииии|иииииииииииии
```

```
·|·····|·····
|  Contract                          ·  Method                                                                  ·  Min        ·  Max        ·  Avg         ·  # calls     ·  usd (avg)
·····|·····|·····|·····|·····|·····|·····
|  ERC725                            ·  transferOwnership(address)                                              ·          -  ·          -  ·       31481  ·            1  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP6KeyManager                    ·  execute(bytes)                                                          ·      44069  ·     164809  ·      103869  ·            6  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20InitAbstract   ·  approve(address,uint256)                                                ·      26062  ·      50984  ·       43759  ·           40  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20InitAbstract   ·  transfer(address,uint256)                                               ·      54828  ·      65662  ·       58890  ·           18  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20InitAbstract   ·  transferFrom(address,address,uint256)                                   ·      55301  ·      66152  ·       59373  ·           18  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20Mintable       ·  mint(address,uint256,bool,bytes)                                        ·      57882  ·      79146  ·       72978  ·          123  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20MintableInit   ·  initialize(string,string,address)                                       ·          -  ·          -  ·      149148  ·           22  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7DigitalAsset                  ·  authorizeOperator(address,uint256)                                      ·      26442  ·      48994  ·       40738  ·          256  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7DigitalAsset                  ·  revokeOperator(address)                                                 ·      23922  ·      26591  ·       25257  ·            4  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7DigitalAsset                  ·  transfer(address,address,uint256,bool,bytes)                            ·      54931  ·      71168  ·       64180  ·           48  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7DigitalAsset                  ·  transferBatch(address[],address[],uint256[],bool,bytes[])               ·      67890  ·     102594  ·       86836  ·           48  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7DigitalAssetInit              ·  initialize(string,string,address,bool)                                  ·     149167  ·     171818  ·      151998  ·           72  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7Mintable                      ·  mint(address,uint256,bool,bytes)                                        ·          -  ·          -  ·       76472  ·            1  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7Mintable                      ·  transferOwnership(address)                                              ·          -  ·          -  ·       28798  ·            1  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile                  ·  setData(bytes32[],bytes[])                                              ·     244260  ·     244296  ·      244278  ·            2  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile                  ·  transferOwnership(address)                                              ·          -  ·          -  ·       46163  ·            2  ·          -
·····|·····|·····|·····|·····|·····|·····
|  Deployments                                                                                                  ·                                    ·  % of limit  ·
·····|·····|·····|·····|·····|·····|·····
|  LSP1UniversalReceiverDelegateUP                                                                              ·          -  ·          -  ·     1511074  ·          5 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP6KeyManager                                                                                               ·          -  ·          -  ·     2446105  ·        8.2 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7Mintable                                                                                                 ·          -  ·          -  ·     1686820  ·        5.6 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7MintableInit                                                                                             ·          -  ·          -  ·     1876750  ·        6.3 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile                                                                                             ·          -  ·          -  ·     2090994  ·          7 %  ·          -
·····|·····|·····|·····|·····|·····|·····

  195 passing (4m)

            should revert
          when caller has permission STATICCALL + 2 x allowed addresses
                    should not allow transfering the tokenId
                when `to` is a contract
                  when receiving contract supports LSP1
             should update lsp5keys (no pop and swap as TokenB has the last index): arrayLength 1, no map, no tokenB address in Vault1
             should keep the same lsp5keys : arrayLength 2, index = 1, tokenB address in Vault2
          When transferring tokenId 1 (all balance) of token C from UP1 to UP2
             should revert when trying to interact with a non-allowed address
            when interacting with 1st allowed contract
               should allow to call view function -> getName()
             should update lsp5keys (no pop and swap as TokenC has the last index): arrayLength 0, no map, no tokenB address in Vault1
             should register lsp5keys : arrayLength 3, index = 2, tokenC address in Vault2
          When transferring 1 tokenId (not all balance) of token B from UP2 to UP1
                      should allow transfering the tokenId
                    when receiving contract does not support LSP1
               should allow to call view function -> getNumber()
                       should not allow transfering the tokenId
             when the from address is incorrect
             should register lsp5keys (UP1 able to re-register keys) : arrayLength 1, index = 0, tokenB address in Vault1
                          should not allow transfering the tokenId
                      when the from address is incorrect
                should register lsp5keys (UP1 able to re-register keys) : arrayLength 1, index = 0, tokenB address in Vault1

·-------------------------------------------------------------------------|---------------------------|--------------|-----------------------------·
|                        Solc version: 0.8.10                             ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
·····|·····|·····|·····|·····|·····|·····
|  Methods
·····|·····|·····|·····|·····|·····|·····
|  Contract                      ·  Method                                 ·  Min        ·  Max        ·  Avg         ·  # calls      ·  usd (avg)
·····|·····|·····|·····|·····|·····|·····
|  LSP6KeyManager                ·  execute(bytes)                         ·      44069  ·     368837  ·      172532  ·           50  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP7CompatibleERC20Mintable   ·  mint(address,uint256,bool,bytes)       ·     165287  ·     207217  ·      183402  ·            6  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP8CompatibleERC721Mintable  ·  mint(address,bytes32,bool,bytes)       ·     130272  ·     274565  ·      204211  ·           10  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP9Vault                     ·  claimOwnership()                       ·          -  ·          -  ·      137332  ·            1  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP9Vault                     ·  transferOwnership(address)             ·          -  ·          -  ·       46141  ·            3  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile              ·  execute(uint256,address,uint256,bytes) ·      59074  ·     313300  ·      164228  ·           34  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile              ·  setData(bytes32[],bytes[])             ·          -  ·          -  ·      244320  ·            6  ·          -
·····|·····|·····|·····|·····|·····|·····
|  UniversalProfile              ·  transferOwnership(address)             ·          -  ·          -  ·       46163  ·            6  ·          -
·····|·····|·····|·····|·····|·····|·····
|  Deployments                                                             ·                                    ·  % of limit   ·
·····|·····|·····|·····|·····|·····|·····
|  LSP1UniversalReceiverDelegateUP                                         ·          -  ·          -  ·     1511074  ·          5 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP1UniversalReceiverDelegateVault                                      ·          -  ·          -  ·     1379449  ·        4.6 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP6KeyManager                                                          ·          -  ·          -  ·     2446105  ·        8.2 %  ·          -
·····|·····|·····|·····|·····|·····|·····
|  LSP9Vault                                                               ·    2000940  ·    2117646  ·     2014237  ·        6.7 %  ·          -
·····|·····|·····|·····|·····|·····|·····
```

```
|  UniversalProfile                                                          и          -  и          -  и     2091006  и          7 %  и          -
и---------------------------------------------------------------------------|-------------|-------------|--------------|---------------|-------------и

  112 passing (4m)

               should revert when calling state changing function -> setName(string)
                   should revert
                when the given tokenId has not been minted
               should revert when calling state changing function -> setNumber(uint256)
            when interacting with 2nd allowed contract
                   should revert
                when function parameters list length does not match
               should allow to interact with 2nd allowed contract - getName()
               should allow to interact with 2nd allowed contract - getNumber()
                   should revert
                when the caller is not an operator
               should revert when calling state changing function -> setName(string)
                   should revert
          _burn
            when tokenId has not been minted
               should revert
            when tokenId has been minted
              after burning a tokenId
               should revert when calling state changing function -> setNumber(uint256)
          when caller has permission SUPER_STATICCALL + 2 allowed addresses
            it should bypass allowed addresses check + allow to interact with any contract
                 should have decreased the total supply
               e.g: Target Contract nb 1
                 should have emitted a Transfer event with address(0) as `to` param
                when calling `tokenOwnerOf(...)` for the burnt tokenId
                   should revert stating tokenId does not exist
                when calling `tokenIdsOf(...)` with the initial owner address of the burnt token
               e.g: Target Contract nb 2
                   should return a list of tokenIds that does not contain the burnt tokenId
                when trying to get the operators for the burnt tokenId
               e.g: Target Contract nb 3
                   should revert stating tokenId does not exist

  LSP8CappedSupply
    when using LSP8CappedSupply with constructor
      when deploying the contract
        when the contract was initialized
           should have registered the ERC165 interface
           should have registered the ERC725Y interface
           should have registered the LSP8 interface
           should have set expected entries with ERC725Y.setData
      when testing deployed contract
        tokenSupplyCap
           should allow reading tokenSupplyCap
        when minting tokens
               e.g: Target Contract nb 4
           should allow minting amount up to tokenSupplyCap
          when cap has been reached
             should error when minting more than tokenSupplyCapTokens
               e.g: Target Contract nb 5
        DELEGATECALL
          when trying to make a DELEGATECALL via UP, DELEGATECALL is disallowed
             should allow minting after burning
    when using LSP8CappedSupply with proxy
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
             should have registered the ERC165 interface
             should revert even if caller has ALL PERMISSIONS
             should have registered the ERC725Y interface
             should have registered the LSP8 interface
             should revert even if caller has permission DELEGATECALL
             should have set expected entries with ERC725Y.setData
        when calling initialize more than once
             should revert with operation disallowed, even if caller does not have permission DELEGATECALL
          when caller has permission SUPER_DELEGATECALL + 2 x allowed addresses
            when calling a disallowed contract
              it should revert since DELEGATECALL is disallowed
           should revert
      when testing deployed contract
        tokenSupplyCap
           should allow reading tokenSupplyCap
        when minting tokens
                 delegate call to contract nb 0
           should allow minting amount up to tokenSupplyCap
          when cap has been reached
             should error when minting more than tokenSupplyCapTokens
                 delegate call to contract nb 1
             should allow minting after burning

  LSP8CompatibleERC721
    when using LSP8CompatibleERC721 contract with constructor
      when deploying the contract
        when initializing the contract
          when the contract was initialized
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have registered its ERC165 interface
                 delegate call to contract nb 2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when checking supported ERC165 interfaces
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should support ERC721
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should support ERC721Metadata
        name
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should allow reading name
        symbol
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should allow reading symbol
        tokenURI
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should allow reading tokenURI
        ownerOf
          when tokenId has not been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when tokenId has been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return owner address
        approve
          when tokenId has not been minted
                 delegate call to contract nb 3
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when the tokenId has been minted
            when caller is not owner of tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert
            when caller is owner of tokenId
              when operator is not the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should succeed
              when operator is the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should revert
        setApprovalForAll
          when calling setApprovalForAll with true
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 delegate call to contract nb 4
            when calling an allowed contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when trying to pass caller address as operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have emitted an ApprovalForAll event
            when calling isApprovedForAll
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert with DELEGATECALL disallowed when trying to interact with the 1st allowed contract
               should return true for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should return false for non-operator
          when operator transfer tokenId
            for tokenId 0xf9cf48dad4314d77852b91dfd2ac169c398ce27d20bfacf4add50af358e743ef:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert with DELEGATECALL disallowed when trying to interact with the 2nd allowed contract
        DEPLOY
          when caller has ALL PERMISSIONS
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to deploy a contract TargetContract via CREATE
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to deploy a contract TargetContract via CREATE2
          when caller is an address with permission DEPLOY
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to deploy a contract TargetContract via CREATE
               should have cleared operators array
            for tokenId 0xd1827dc04a206e983a40a7e7fa9d3fbaddc269d95dc07afd94808febcf7c7fda:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
             should be allowed to deploy a contract TargetContract via CREATE2
          when caller is an address that does not have the permission DEPLOY
            when calling via execute(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert when trying to deploy a contract via CREATE
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert when trying to deploy a contract via CREATE2
            when calling via executeRelayCall(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert when trying to deploy a contract via CREATE
               should have cleared operators array
            for tokenId 0xf4f30a065b7a210c0b81cc9be61666043cbd7a6c576fb1cd8022a532a14610b5:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert when trying to deploy a contract via CREATE2
        TRANSFERVALUE
          when caller = EOA
            when recipient = EOA
              when transferring value without bytes `_data`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when caller has ALL PERMISSIONS
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when caller has permission TRANSFERVALUE only
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when caller has permission TRANSFERVALUE + CALL
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should fail when caller does not have permission TRANSFERVALUE
              when transferring value with bytes `_data`
               should have cleared operators array
            for tokenId 0x24248d4bb005cdd61e85488ec603a8f8ac3f1a5bc1af6ca1ca5e29bf2f39285d:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when caller has ALL PERMISSIONS
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when caller has permission TRANSFERVALUE + CALL
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should fail when caller has permission TRANSFERVALUE only
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should fail when caller does not have permission TRANSFERVALUE
          when caller = contract
            > Contract calls
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Should send 1 LYX to an address hardcoded in Executor (`sendOneLyxHardcoded`)
               should have cleared operators array
            for tokenId 0xb0c09f1a7a4b027ad28c7a4be659e9a18d1534a54bab492838fc09def0d9e4d2:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Should send 1 LYX to an address provided to Executor (`sendOneLyxToRecipient`)
            > Low-level calls
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Should send 1 LYX to an address hardcoded in Executor (`sendOneLyxHardcodedRawCall`)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Should send 1 LYX to an address provided to Executor (`sendOneLyxToRecipientRawCall`)
          when caller is another UP (with a KeyManager as owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have cleared operators array
          when calling setApprovalForAll with false (removing operator full approval)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Alice should have ALL PERMISSIONS in her UP
             Bob should have ALL PERMISSIONS in his UP
             Alice's UP should have permission TRANSFERVALUE on Bob's UP
             Alice should be able to send 5 LYX from Bob's UP to her UP
          when caller has SUPER_TRANSFERVALUE + CALL
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should not be allowed to interact with a disallowed LSP7 contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xf9cf48dad4314d77852b91dfd2ac169c398ce27d20bfacf4add50af358e743ef with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to interact with an allowed LSP7 contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xd1827dc04a206e983a40a7e7fa9d3fbaddc269d95dc07afd94808febcf7c7fda with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to interact with an allowed contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to interact with an allowed contract + send some LYX while calling the function
            should be allowed to send LYX to any EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xf4f30a065b7a210c0b81cc9be61666043cbd7a6c576fb1cd8022a532a14610b5 with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
               should send LYX to EOA -> 0xD782F87eB01763bB813f2571400a71D4c0092784
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0x24248d4bb005cdd61e85488ec603a8f8ac3f1a5bc1af6ca1ca5e29bf2f39285d with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0xfC2b37e1e137d90f50f1eF888EFFBd1056651ce2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0xCEE510FB8BAf5Aa31449d8235A4461f3693A98F3
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xb0c09f1a7a4b027ad28c7a4be659e9a18d1534a54bab492838fc09def0d9e4d2 with transferFrom(...)
        getApproved
          when tokenId has not been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when tokenId has been minted
            when there have been no approvals for the tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should return address(0)
            when one account has been approved for the tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0x73d8c8ACcB308bff1f49C37E60b64b00583dE179
               should return the operator address
            when many context.accounts have been approved for the tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should return the last new authorized operator
        mint
          when a token is minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have expected events
        burn
          when a token is burned
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0xf0dcA2300d5FE6738E00d4fe934265F66Ba7344d
            should be allowed to send LYX to any other UP contract
             should have expected events
        transfers
          transferFrom
            when the from address is the tokenId owner
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to UP 0
                 should allow transfering the tokenId
              when `to` is a contract
                when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to UP 1
                   should allow transfering the tokenId
            when the from address is not the tokenId owner
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to UP 2
               should revert
          safeTransferFrom(address,address,uint256)
            when the from address is the tokenId owner
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should revert
              when `to` is a contract
                when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to UP 3
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   should revert
            when the from address is not the tokenId owner
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to UP 4
          when caller has TRANSFERVALUE + SUPER_CALL
               should revert
          safeTransferFrom(address,address,uint256,bytes)
            when the from address is the tokenId owner
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should revert
              when `to` is a contract
                when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should not be allowed to do a plain LYX transfer to a non-allowed address
                   should allow transfering the tokenId
                when receiving contract does not support LSP1
             should be allowed to do a plain LYX transfer to an allowed address
            should be allowed to interact with any contract
              eg: any TargetContract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   should revert
            when the from address is not the tokenId owner
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 1
               should revert
    when using LSP8 contract with proxy
      when deploying the base implementation contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
         prevent any address from calling the initialize(...) function on the implementation
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have registered its ERC165 interface
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have set expected entries with ERC725Y.setData
        when calling initialize more than once
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 2
           should revert
      when testing deployed contract
        when checking supported ERC165 interfaces
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should support ERC721
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 3
           should support ERC721Metadata
        name
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should allow reading name
        symbol
                 TargetContract nb 4
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           should allow reading symbol
        tokenURI
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
                 TargetContract nb 5
              eg: any LSP7 Token owned by the UP
           should allow reading tokenURI
        ownerOf
          when tokenId has not been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when tokenId has been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 1
             should return owner address
        approve
          when tokenId has not been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when the tokenId has been minted
            when caller is not owner of tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert
            when caller is owner of tokenId
              when operator is not the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 3
                 should succeed
              when operator is the zero address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 4
                 should revert
        setApprovalForAll
          when calling setApprovalForAll with true
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 5
            should not be allowed to interact with any contract if sending LYX along the call
               Target Payable Contract nb 1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when trying to pass caller address as operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Target Payable Contract nb 2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Target Payable Contract nb 3
               Target Payable Contract nb 4
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have emitted an ApprovalForAll event
            when calling isApprovedForAll
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Target Payable Contract nb 5
          when caller has SUPER_TRANSFERVALUE + SUPER_CALL
            should be allowed to send LYX to any address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0x32520d84dE24357D70bd6f77D8fF506d9187F23f
               should send LYX to EOA -> 0xbCb4c3448b5fAf31cB85438C8F1c9a0dF43c9178
               should return true for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0x76b04187a14524D1E3F1031CeE16f7fA44e5e479
               should send LYX to EOA -> 0xF8d262804Fe39863D0b3cdD8fcAbAAd5A862f3aC
               should return false for non-operator
          when operator transfer tokenId
            for tokenId 0xf9cf48dad4314d77852b91dfd2ac169c398ce27d20bfacf4add50af358e743ef:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should send LYX to EOA -> 0xb0D3c06CB9d55A8560EFdb24B846b2D57C1c42EE
            should be allowed to interact with any contract
              eg: any TargetContract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 1
                 TargetContract nb 2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 3
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 TargetContract nb 4
                 TargetContract nb 5
              eg: any LSP7 Token owned by the UP
                 LSP7DigitalAsset nb 1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 2
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 3
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 4
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have cleared operators array
            for tokenId 0xd1827dc04a206e983a40a7e7fa9d3fbaddc269d95dc07afd94808febcf7c7fda:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 LSP7DigitalAsset nb 5
        SIGN (ERC1271)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
           can verify signature from address with ALL PERMISSIONS on KeyManager
           can verify signature from signer on KeyManager
           should fail when verifying signature from address with no SIGN permission
           should fail when verifying signature from address with no permissions set
        ALLOWEDADDRESSES
          when caller has no ALLOWED ADDRESSES set
            it should be allowed to interact with any address
               sending 1 LYX to EOA 0x2a22ac2b4b83d39b4b0bf4a914e9524268435bec
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0x6ff70b24c8ffbd929e7963f7298f6e6c7507f00b
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0x13fda878395661cf49dc08204450e66dc611e601
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0xf85defe6be667d371135893b71405dc567fce2fb
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0xff6610a3489d15028e670cadd3bbdd321b722fe0
          when caller has 2 x ALLOWED ADDRESSES set
             should be allowed to send LYX to an allowed address (= EOA)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should be allowed to interact with an allowed address (= contract)
               should have cleared operators array
            for tokenId 0xf4f30a065b7a210c0b81cc9be61666043cbd7a6c576fb1cd8022a532a14610b5:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when sending LYX to a non-allowed address (= EOA)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when interacting with an non-allowed address (= contract)
          when caller has an invalid abi-encoded array set for ALLOWED ADDRESSES
            it should be allowed to interact with any address
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0xeb02a8bced64718bdaa8a25398f21f82816b9085
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0x812395f73bbfa8c8ab3c8a81f8607f32fcd1b332
               sending 1 LYX to EOA 0x0c37d00dfb72ca1de494b68768c505d428434b9c
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have emitted a Transfer event
               sending 1 LYX to EOA 0xf757cf192424e1731a9b30198478cdfb3553f86d
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               sending 1 LYX to EOA 0x0647910240dba2c838cbeffad24e88aa0e3f235a
        ALLOWEDFUNCTIONS
          when interacting via `execute(...)`
            when caller has nothing listed under AllowedFunctions
              when calling a contract
                 should pass when calling any function (eg: `setName(...)`)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when calling any function (eg: `setNumber(...)`)
            when caller has 1 x bytes4 function selector listed under AllowedFunctions
               should have cleared operators array
            for tokenId 0x24248d4bb005cdd61e85488ec603a8f8ac3f1a5bc1af6ca1ca5e29bf2f39285d:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should revert when passing a random bytes payload with a random function selector
              when calling a contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when the bytes4 selector of the function called is listed in its AllowedFunctions
                 should revert when the bytes4 selector of the function called is NOT listed in its AllowedFunctions
          when interacting via `executeRelayCall(...)`
            when signer has 1 x bytes4 function selector listed under AllowedFunctions
              when calling a contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))

Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 `setName(...)` - should pass when the bytes4 selector of the function called is listed in its AllowedFunctions
                 `setNumber(...)` - should revert when the bytes4 selector of the function called is NOT listed in its AllowedFunctions
        ALLOWEDSTANDARDS
          when caller has no value set for ALLOWEDSTANDARDS (= all interfaces whitelisted)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should allow to interact with contract that does not implement any interface
            should allow to interact with a contract that implement (+ register) any interface
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               ERC1271
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               LSP0 (ERC725Account)
          when caller has only ERC1271 interface ID set for ALLOWED STANDARDS
            when interacting with a contract that implements + register ERC1271 interface
               should have cleared operators array
            for tokenId 0xb0c09f1a7a4b027ad28c7a4be659e9a18d1534a54bab492838fc09def0d9e4d2:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should pass
            when trying to interact an ERC725Account (LSP0)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should allow to transfer LYX
            when interacting with contract that does not implement ERC1271
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should have transferred successfully with `transferFrom(...)` (changed token owner)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should fail
          when caller has only LSP7 interface ID set for ALLOWED STANDARDS
            when interacting with a contract that implements + register ERC1271 interface
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should fail
            when interacting with an ERC725Account (LSP0)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should fail when trying to transfer LYX
        ALLOWEDERC725YKeys
          keyType: Singleton
               should have emitted a Transfer event
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
            verify allowed ERC725Y keys set
               `controllerCanSetOneKey` should have 1 x key in its list of allowed keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               `controllerCanSetManyKeys` should have 3 x keys in its list of allowed keys
               `controllerCanSetOneKey` should have the right keys set in its list of allowed keys
               `controllerCanSetManyKeys` should have the right keys set in its list of allowed keys
            when address can set only one key
              when setting one key
                 should pass when setting the right key
                 should fail when setting the wrong key
              when setting multiple keys
                 should fail when the list contains none of the allowed keys
                 should fail, even if the list contains the allowed key
            when address can set multiple keys
               should pass when the input is all the allowed keys
               should fail when the input contains none of the allowed keys
              when setting one key
                 should pass when trying to set the 1st allowed key
                 should pass when trying to set the 2nd allowed key
                 should pass when trying to set the 3rd allowed key
                 should fail when setting a not-allowed Singleton key
              when setting 2 x keys
                should pass when
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   the input is the first two (subset) allowed keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   the input is the last two (subset) allowed keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   the input is the first + last (subset) allowed keys
              when setting 3 x keys
                should fail when
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   1st key in input = 1st allowed key. Other 2 keys = not allowed
                   2nd key in input = 1st allowed key. Other 2 keys = not allowed
                   3rd key in input = 1st allowed key. Other 2 keys = not allowed
                   1st key in input = 2nd allowed key. Other 2 keys = not allowed
                   2nd key in input = 2nd allowed key. Other 2 keys = not allowed
                   3rd key in input = 2nd allowed key. Other 2 keys = not allowed
                   1st key in input = 3rd allowed key. Other 2 keys = not allowed
                   2nd key in input = 3rd allowed key. Other 2 keys = not allowed
               should have cleared operators array
          when calling setApprovalForAll with false (removing operator full approval)
                   3rd key in input = 3rd allowed key. Other 2 keys = not allowed
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                   1st key in input = not allowed key. Other 2 keys = allowed
                   2nd key in input = not allowed key. Other 2 keys = allowed
                   3rd key in input = not allowed key. Other 2 keys = allowed
              when setting multiple keys
                when input is bigger than the number of allowed keys
                  should fail when
                     input = all the allowed keys + 1 x not-allowed key
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                     input = all the allowed keys + 5 x not-allowed key
                  should pass when
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                     input contains all the allowed keys as DUPLICATE
            when address can set any key
              when setting one key
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when setting any random key
              when setting multiple keys
                 should pass when setting any multiple keys
          keyType: Mapping
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
            when address can set Mapping keys starting with a 'SupportedStandards:...'
              when setting one key
                 should pass when setting SupportedStandards:LSPX
                 should pass when overriding SupportedStandards:LSPX
                 should pass when setting SupportedStandards:LSPY
                 should pass when setting SupportedStandards:LSPZ
                 should fail when setting any other not-allowed Mapping key
              when setting multiple keys
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 (2 x keys) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xf9cf48dad4314d77852b91dfd2ac169c398ce27d20bfacf4add50af358e743ef with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 (2 x keys) (override) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
                 (3 x keys) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 (3 x keys) (override) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
                 should fail when the list contains none of the allowed Mapping keys
                 should fail, even if the list contains some keys starting with `SupportedStandards`
            when address can set any key
              when setting one key
                 should pass when setting any random Mapping key
              when setting multiple keys
                 should pass when setting any random set of Mapping keys
          keyType: Array
            when address can set Array element in 'MyArray[]
              when setting one key
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when setting array key length MyArray[]
             should return false when calling isApprovedForAll for operator
                 should pass when setting 1st array element MyArray[0]
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should pass when setting 2nd array element MyArray[1]
```

```
                 should pass when setting 3rd array element MyArray[3]
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                 should fail when setting elements of a not-allowed Array (eg: LSP5ReceivedAssets)
              when setting multiple keys
test gas cost:  82343
                 should pass when all the keys in the list are from the allowed array MyArray[]
                 should fail when the list contains elements keys of a non-allowed Array (RandomArray[])
                 should fail, even if the list contains a mix of allowed + not-allowed array element keys (MyArray[] + RandomArray[])
          bytes32(0) (= zero key) as one of the allowed ERC725Y data key
             should pass when setting a data key listed in the allowed ERC725Y data keys: 0x7da57cc766f1ea8499381d54a05adf511288c2b3d84a1fee58d723eb84772f
70
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xd1827dc04a206e983a40a7e7fa9d3fbaddc269d95dc07afd94808febcf7c7fda with transferFrom(...)
             should pass when setting a data key listed in the allowed ERC725Y data keys: 0x201bc59d8e67d071c9403301d56347067ad8f4de28b0bf41dcfc265d1662bc
d4
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should pass when setting a data key listed in the allowed ERC725Y data keys: 0x00000000000000000000000000000000000000000000000000000000000000
00
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should pass when trying to set any random data key (e.g: 0x07f83b904e398f70e60c2275e5e7b4372d1ebfaad436d5d23ff03c195d29c5ed)
             should pass when trying to set any random data key (e.g: 0xb30719f9be7ce927acc48128e57983c34807c6a0c2ec320fa0e485708b57a16d)
             should pass when trying to set any random data key (e.g: 0xe08af8ae29b8c0392e085d8cb170b52c14487abed75cc1107c0fc031feb856a7)
             should pass when trying to set any random data key (e.g: 0xe6d0d73826694058b1dab02bab34777916e23570276506b5d29c58408f366d40)
             should pass when trying to set any random data key (e.g: 0xe04c8e802ce454c81493e24a1e037b26b1de4fd24ab619cd3bb6d51bf34825e8)
          one single byte as an allowed data key (e.g: 0xaa0000...0000
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
            should pass when setting a data key that starts with `0xaa`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               e.g: 0xaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               e.g: 0xaacccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
               e.g: 0xaadddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
              when trying to set a data key that does not start with `0xaa`
                 should revert (e.g: 0xbb00000000000000000000000000000000000000000000000000000000000000)
                 should revert (e.g: 0xcc00000000000000000000000000000000000000000000000000000000000000)
                 should revert (e.g: 0xdd00000000000000000000000000000000000000000000000000000000000000)
                 should revert (e.g: 0xbb12345678000000000000000000000000000000000000000000000000000000)
                 should revert (e.g: 0xcc12345678000000000000000000000000000000000000000000000000000000)
                 should revert (e.g: 0xdd12345678000000000000000000000000000000000000000000000000000000)
        Multi Channel nonces
          testing sequential nonces (channel = 0)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xf4f30a065b7a210c0b81cc9be61666043cbd7a6c576fb1cd8022a532a14610b5 with transferFrom(...)
             First call > nonce should increment from undefined to NaN
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Second call > nonce should increment from undefined to NaN
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Third call > nonce should increment from undefined to NaN
             Fourth call > nonce should increment from undefined to NaN
          out of order execution (channel = n)
            channel 1
               First call > nonce should increment from 0 to 1
               Second call > nonce should increment from 1 to 2
            channel 2
               First call > nonce should increment from 0 to 1
               Second call > nonce should increment from 1 to 2
            channel 3
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
               First call > nonce should increment from 0 to 1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Second call > nonce should increment from 1 to 2
            channel 15
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               First call > nonce should increment from 0 to 1
        miscellaneous
          payload
            - should fail when sending an empty payload to `keyManager.execute('0x')`
             Should revert because calling an unexisting function in ERC725
          wrong operation type
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Should revert because of wrong operation type when caller has ALL PERMISSIONS
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0x24248d4bb005cdd61e85488ec603a8f8ac3f1a5bc1af6ca1ca5e29bf2f39285d with transferFrom(...)
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Should revert because of wrong operation type when caller has not ALL PERMISSIONS
        Security
           Should revert when caller has no permissions set
          should revert when admin with ALL PERMISSIONS try to call `renounceOwnership(...)`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return false when calling isApprovedForAll for operator
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             via `execute(...)`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             via `executeRelayCall()`
          when sending LYX to a contract
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             Permissions should prevent ReEntrancy and stop malicious contract with a re-entrant fallback() function.
            when calling via `executeRelayCall(...)`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert when operator try to transfer tokenId 0xb0c09f1a7a4b027ad28c7a4be659e9a18d1534a54bab492838fc09def0d9e4d2 with transferFrom(...)
        getApproved
          when tokenId has not been minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should revert
          when tokenId has been minted
            when there have been no approvals for the tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               Replay Attack should fail because of invalid nonce
      testing internal functions
               should return address(0)
            when one account has been approved for the tokenId
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
```

```
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
         should return the right list of allowed functions
               should return the operator address
            when many context.accounts have been approved for the tokenId
         should return an empty byte when address has no allowed functions listed
        testing abi-encoded array of `address[]`
          `getAllowedAddressesFor(...)`
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
               should return the last new authorized operator
        mint
          when a token is minted
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return the same list of allowed addresses
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should have expected events
        burn
          when a token is burned
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should return no bytes when no allowed addresses are set
          `verifyAllowedAddressesFor(...)`
             should have expected events
        transfers
          transferFrom
            when the from address is the tokenId owner
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should not revert for an address listed in allowed addresses list
                 should allow transfering the tokenId
               when `to` is a contract
                 when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
              should revert for address not listed in allowed addresses list
                   should allow transfering the tokenId
                 when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
             should not revert when user has no address listed (= all addresses whitelisted)
         testing 'zero bytes' stored under AddressPermission:AllowedAddresses:<address>
                   should allow transfering the tokenId
            when the from address is not the tokenId owner
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
          `verifyAllowedAddressesFor(...)`
            should not revert and consider the stored value as all addresses whitelisted for:
               should revert
          safeTransferFrom(address,address,uint256)
            when the from address is the tokenId owner
              when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                noBytes -> 0x
                oneZeroByte -> 0x00
                  should revert
               when `to` is a contract
                 when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                tenZeroBytes -> 0x00000000000000000000
                twentyZeroBytes -> 0x0000000000000000000000000000000000000000
            should revert with NotAllowedAddress(...) error for:
                   should allow transfering the tokenId
                 when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                thirtyTwoZeroBytes -> 0x0000000000000000000000000000000000000000000000000000000000000000
                   should revert
            when the from address is not the tokenId owner
                fourtyZeroBytes -> 0x00000000000000000000000000000000000000000000000000000000000000000000000000000000
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                sixtyFourZeroBytes -> 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000
                should revert
           safeTransferFrom(address,address,uint256,bytes)
             when the from address is the tokenId owner
               when `to` is an EOA
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                hundredZeroBytes -> 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
        testing random values under the key `AddressPermissions:AllowedAddress:<address>`
                 should revert
               when `to` is a contract
                 when receiving contract supports LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
          `verifyAllowedAddressesFor(...)`
            should revert with NotAllowedAddress(...) error for:
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                emptyABIEncodedArray -> 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
000000000000000
                emptyABIEncodedArrayWithMoreZeros -> 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
                   should allow transfering the tokenId
                 when receiving contract does not support LSP1
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                multipleOf32Bytes -> 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000
0000000000010000000000000000000000000a0Ee7A142d267C1f36714E4a8F75612F20a79720
            should not revert and consider the stored value as all addresses whitelisted for:
               shortBytes -> 0xaabbccdd
               longBytes -> 0x1234567890abcdef1234567890abcdef
        keyType: Singleton
          getAllowedERC725YKeysFor(...)
                   should revert
            when the from address is not the tokenId owner
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
Duplicate definition of Transfer (Transfer(address,address,uint256), Transfer(address,address,address,bytes32,bool,bytes))
                should revert

  LSP8Enumerable
    when using LSP8Enumerable with constructor
      when deploying the contract
        when the contract was initialized
           should have registered the ERC165 interface
             should return the same list of allowed ERC725Y Keys
          verifyAllowedERC725YKeys(...)
           should have registered the ERC725Y interface
           should have registered the LSP8 interface
           should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when no minted tokens
           should not get token
        when minted tokens
           should access by index
             should revert even if list contains one allowed key
        _countZeroBytes(...)
          test against LSP2 key types
           should not access by index after removed
           should access by index after removed
    when using LSP8Enumerable with proxy
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
             should have registered the ERC165 interface
             Singleton: should return 0 for `LSP3Profile` -> 0x5ef83ad9559033e6e941db7d7c495acdce616347d28e90c7ce47cbfcfcad3bc5
             should have registered the ERC725Y interface
             should have registered the LSP8 interface
             Array: should return 16 for `LSP4Creators[]` -> 0x114bd03b3a46d48759680d81ebb2b414000000000000000000000000000000000000
             should have set expected entries with ERC725Y.setData
        when calling initialize more than once
           should revert
      when testing deployed contract
        when no minted tokens
             Mapping: should return 16 for `SupportedStandards:...` -> 0xeafec4d89fa9619884b60000abe425d600000000000000000000000000000000000000
```

```
          should not get token
       when minted tokens
            Bytes20Mapping: should return 16 for `LSP5ReceivedAssetsMap:...` -> 0x812c4334633eb816000000000000000000000000000000000000000000000000
       `getPermissionsFor(...)` -> reading permissions
          should access by index
          Should return ALL_PERMISSIONS for owner
          should not access by index after removed
          Should return SETDATA
          should access by index after removed

  LSP8Mintable
    when using LSP8Mintable with constructor
      when deploying the contract
        when the contract was initialized
          should have registered the ERC165 interface
          should have registered the ERC725Y interface
          should have registered the LSP8 interface
          Should return SETDATA + CALL
        `getPermissionsFor(...)` -> reading empty permissions
          should have set expected entries with ERC725Y.setData
      when testing deployed contract
        when owner minting tokens
          total supply should have increased
          tokenReceiver balance should have increased
        when non-owner minting tokens
          should revert
        when owner try to re-enter function through the UniversalReceiverDelegate
          should cast permissions to 32 bytes when reading permissions stored as more than 32 empty bytes
          should cast permissions to 32 bytes when reading permissions stored as less than 32 empty bytes
          should pass
    when using LSP8Mintable with proxy
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
            should have registered the ERC165 interface
          should cast permissions to 32 bytes when reading permissions stored as one empty byte
        `includesPermissions(...)`
            should have registered the ERC725Y interface
            should have registered the LSP8 interface
            should have set expected entries with ERC725Y.setData
        when calling initialize more than once
          Should return true when checking if has permission SETDATA
        AddressPermissions[]
          should revert
      when testing deployed contract
        when owner minting tokens
          total supply should have increased
          tokenReceiver balance should have increased
        when non-owner minting tokens
          should revert
        when owner try to re-enter function through the UniversalReceiverDelegate
          Value should be 5 for key 'AddressPermissions[]'
          Checking address (=value) stored at AddressPermissions[1]'
          Checking address (=value) stored at AddressPermissions[2]'
          should pass

·-------------------------------------------------------------------------------------------|--------------------------|-------------|-----------------·
|                                  Solc version: 0.8.10                                     ·  Optimizer enabled: true  ·  Runs: 1000  ·  Block limit: 30000000 gas
···········································································|··························|·············|·····························|·····························
|  Methods

···········································································|··························|·············|·····························|·····························
|  Contract                        ·  Method                                                 ·  Min        ·  Max        ·  Avg         ·  # calls      ·  usd (avg)
···········································································|··························|·············|·····························|·····························
|  ERC725                          ·  transferOwnership(address)                             ·          -  ·          -  ·       31503  ·  1          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP6KeyManager                  ·  execute(bytes)                                         ·      44069  ·     299419  ·      148741  ·  6          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP7CompatibleERC20InitAbstract ·  approve(address,uint256)                               ·      80532  ·     100746  ·       93466  ·  66         ·          -
···········································································|··························|·············|·····························|·····························
|  LSP7CompatibleERC20InitAbstract ·  transferFrom(address,address,uint256)                  ·     113184  ·     148476  ·      126628  ·  64         ·          -
···········································································|··························|·············|·····························|·····························
|  LSP7CompatibleERC20MintableInit ·  initialize(string,string,address)                      ·     146729  ·     169272  ·      151143  ·  107        ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8CompatibleERC721InitAbstract·  safeTransferFrom(address,address,uint256,bytes)        ·     122831  ·     125552  ·      124192  ·  8          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8CompatibleERC721InitAbstract·  safeTransferFrom(address,address,uint256)             ·     120913  ·     123584  ·      122249  ·  8          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8CompatibleERC721InitAbstract·  setApprovalForAll(address,bool)                       ·      24309  ·      48918  ·       41254  ·  82         ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8CompatibleERC721Mintable    ·  mint(address,bytes32,bool,bytes)                       ·     105374  ·     146452  ·      133495  ·  193        ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8IdentifiableDigitalAsset    ·  authorizeOperator(address,bytes32)                     ·      76080  ·      95855  ·       86337  ·  276        ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8IdentifiableDigitalAsset    ·  revokeOperator(address,bytes32)                        ·      33707  ·      35828  ·       34768  ·  4          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8IdentifiableDigitalAsset    ·  transfer(address,address,bytes32,bool,bytes)           ·     122433  ·     133815  ·      128511  ·  72         ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8IdentifiableDigitalAsset    ·  transferBatch(address[],address[],bytes32[],bool,bytes[])·     208096  ·     225245  ·      216671  ·  112        ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8Mintable                    ·  mint(address,bytes32,bool,bytes)                       ·          -  ·          -  ·      143778  ·  1          ·          -
···········································································|··························|·············|·····························|·····························
|  LSP8Mintable                    ·  transferOwnership(address)                             ·          -  ·          -  ·       28823  ·  1          ·          -
···········································································|··························|·············|·····························|·····························
|  UniversalProfile                ·  setData(bytes32[],bytes[])                             ·          -  ·          -  ·      244296  ·  2          ·          -
···········································································|··························|·············|·····························|·····························
|  UniversalProfile                ·  transferOwnership(address)                             ·          -  ·          -  ·       46163  ·  2          ·          -
···········································································|··························|·············|·····························|·····························
|  Deployments                                                                               ·                                       ·  % of limit   ·
···········································································|··························|·············|·····························|·····························
|  LSP1UniversalReceiverDelegateUP                                                           ·          -  ·          -  ·     1511074  ·  5          %  ·          -
···········································································|··························|·············|·····························|·····························
|  LSP6KeyManager                                                                            ·          -  ·          -  ·     2446105  ·  8.2        %  ·          -
···········································································|··························|·············|·····························|·····························
```

```
|  LSP8Mintable                                                                                     и           -  и           -  и     1986503  и        6.6
%  и           -
ииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|ииииииииииииии|иииииииииииии
иии|иииииииииииии
|  LSP8MintableInit                                                                                 и           -  и           -  и     2165756  и        7.2
%  и           -
ииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|ииииииииииииии|иииииииииииии
иии|иииииииииииии
|  UniversalProfile                                                                                 и           -  и           -  и     2090994  и          7
%  и           -
и-------------------------------------------------------------------------------------------------|-------------|-------------|--------------|------------
---|-------------и

  340 passing (5m)

           Checking address (=value) stored at AddressPermissions[3]'
           Checking address (=value) stored at AddressPermissions[4]'
           Checking address (=value) stored at AddressPermissions[5]'
    when using LSP6KeyManager with proxy
      when deploying the contract as proxy
        when initializing the contract
          when the contract was initialized
             should support ERC165 interface
             should support ERC1271 interface
             should support LSP6 interface
             should be linked to the right ERC725 account contract
        when calling `initialize(...) more than once`
           should revert
      when testing deployed contract
        CHANGEOWNER
          when upgrading to a new KeyManager via transferOwnership(...)
            when caller does not have have CHANGEOWNER permission
               should revert
            when caller has ALL PERMISSIONS
               should have set newKeyManager as pendingOwner
               owner should remain the current KeyManager
               should override the pendingOwner when transferOwnership(...) is called twice
              it should still be possible to call onlyOwner functions via the old KeyManager
                 setData(...)
                 execute(...) - LYX transfer
            when caller has only CHANGE0OWNER permission
               should have set newKeyManager as pendingOwner
               owner should remain the current KeyManager
               should override the pendingOwner when transferOwnership(...) is called twice
          when calling claimOwnership(...) from a KeyManager that is not the pendingOwner
             should revert
          when calling claimOwnership(...) via the pending new KeyManager
             should have change the account's owner to the pendingOwner (= pending KeyManager)
             should have cleared the pendingOwner after transfering ownership
          after KeyManager has been upgraded via claimOwnership(...)
            old KeyManager should not be allowed to call onlyOwner functions anymore
               should revert when calling `setData(...)`
               should revert when calling `execute(...)`
            new Key Manager should be allowed to call onlyOwner functions
               setData(...)
               execute(...) - LYX transfer
        CHANGE / ADD permissions
          setting permissions keys (CHANGE vs ADD Permissions)
            when setting one permission key
              when caller is an address with ALL PERMISSIONS
                 should be allowed to ADD a permission
                 should be allowed to CHANGE a permission
                 should be allowed to increment the 'AddressPermissions[]' key (length)
                 should be allowed to decrement the 'AddressPermissions[]' key (length)
                 should be allowed to edit key at index -> AddressPermissions[4]
                if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
                   should revert
              when caller is an address with permission ADDPERMISSIONS
                 should be allowed to add a permission
                 should not be allowed to CHANGE a permission
                 should be allowed to increment the 'AddressPermissions[]' key (length)
                 should not be allowed to decrement the 'AddressPermissions[]' key (length)
                 should not be allowed to edit key at index -> AddressPermissions[4]
                if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
                   should revert when trying to set a non-standard LSP6 permission data key
              when caller is an address with permission CHANGEPERMISSION
                 should not be allowed to ADD a permission
                 should not be allowed to set (= ADD) a permission for an address that has 32 x 0 bytes (0x0000...0000) as permission value
                 should be allowed to CHANGE a permission
                 should not be allowed to increment the 'AddressPermissions[]' key (length)
                 should be allowed to decrement the 'AddressPermissions[]' key (length)
                 should be allowed to edit key at index -> AddressPermissions[4]
                if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
                   should revert when trying to set a non-standard LSP6 permission data key
              when caller is an address with permission SETDATA
                 should not be allowed to ADD a permission
                 should not be allowed to set (= ADD) a permission for an address that has 32 x 0 bytes (0x0000...0000) as permission value
                 should not be allowed to CHANGE a permission
                 should not be allowed to increment the 'AddressPermissions[]' key (length)
                 should not be allowed to decrement the 'AddressPermissions[]' key (length)
                 should not be allowed to edit key at index -> AddressPermissions[4]
                if the data key starts with AddressPermissions: but is a non-standard LSP6 permission data key
                   should revert when trying to set a non-standard LSP6 permission data key
          setting Allowed Addresses
            when caller has permission ADDPERMISSIONS
               should fail when trying to edit existing allowed addresses for an address
               should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of address[] initially
               should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed addresses
               should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed addresses
               should pass when beneficiary had no values set under AddressPermissions:AllowedAddresses:... + setting a valid abi-encoded array of address
[] (= with 12 x leading '00')
              when setting an invalid abi-encoded array of address[] for a new beneficiary
                 should revert with error when value = random bytes
                 should revert with error when value = invalid abi-encoded array of address[] (not enough leading zero bytes for an address -> 10 x '00')
            when caller has permission CHANGEPERMISSIONS
               should fail when beneficiary had no values set under AddressPermissions:AllowedAddresses:...
               should pass when trying to edit existing allowed addresses for an address
               should pass when address had an invalid abi-encoded array of address[] initially
               should pass when address had 32 x 0 bytes set initially as allowed addresses
               should pass when address had 40 x 0 bytes set initially as allowed addresses
              when changing the list of allowed address of existing address to an invalid value
                 should revert with error when value = random bytes
                 should revert with error when value = invalid abi-encoded array of address[] (not enough leading zero bytes for an address -> 10 x '00')
          setting Allowed Functions
            when caller has permission ADDPERMISSIONS
               should fail when trying to edit existing allowed functions for an address
               should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of bytes4[] initially
               should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed functions
               should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed functions
               should pass when beneficiary had no values set under AddressPermissions:AllowedFunctions:... + setting a valid abi-encoded array of bytes4[
] (= 28 leading zeros)
              when setting an invalid abi-encoded array of bytes4[] selector for a new beneficiary
                 should fail when setting an invalid abi-encoded array of bytes4[] (= random bytes)
                 should fail when setting an invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00')
            when caller has CHANGEPERMISSIONS
               should fail when beneficiary had no values set under AddressPermissions:AllowedFunctions:...
               should pass when trying to edit existing allowed bytes4 selectors for an address
               should pass when address had an invalid abi-encoded array of bytes4[] initially
               should pass when address had 32 x 0 bytes set initially as allowed functions
               should pass when address had 40 x 0 bytes set initially as allowed functions
              when changing the list of allowed bytes4 function selectors to an invalid value
                 should revert with error when value = random bytes
                 should revert with error when value = invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00
')
          setting Allowed Standards
            when caller has ADDPERMISSIONS
               should fail when trying to edit existing allowed standards for an address
               should fail with NotAuthorised -> when beneficiary address had an invalid abi-encoded array of bytes4[] initially
               should fail with NotAuthorised -> when beneficiary had 32 x 0 bytes set initially as allowed functions
               should fail with NotAuthorised -> when beneficiary had 40 x 0 bytes set initially as allowed functions
               should pass when beneficiary had no values set under AddressPermissions:AllowedStandards:... + setting a valid abi-encoded array of bytes4[
] (= 28 leading zeros)
              when setting an invalid abi-encoded array of bytes4[] interface IDs for a new beneficiary
                 should fail when setting an invalid abi-encoded array of bytes4[] (= random bytes)
                 should fail when setting an invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00')
            when caller has CHANGEPERMISSION
               should fail when beneficiary had no values set under AddressPermissions:AllowedStandards:...
               should pass when trying to edit existing allowed standards for an address
               should pass when address had an invalid abi-encoded array of bytes4[] interface IDs initially
               should pass when address had 32 x 0 bytes set initially as allowed standards
```

```
               should pass when address had 40 x 0 bytes set initially as allowed standards
            when changing the list of allowed bytes4 interface IDs to an invalid value
               should revert with error when value = random bytes
               should revert with error when value = invalid abi-encoded array of bytes4[] (not enough leading zero bytes for a bytes4 value -> 26 x '00
')
         setting Allowed ERC725YKeys
           when caller has ADDPERMISSIONS
             when beneficiary had some ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
               should fail when adding an extra allowed ERC725Y data key
               should fail when removing an allowed ERC725Y data key
               should fail when trying to clear the array completely
               should fail when setting an invalid abi-encoded array of bytes32[]
             when beneficiary had no ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
               should pass when setting a valid abi-encoded array of bytes32[]
               should fail when setting an invalid abi-encoded array of bytes32[] (random bytes)
           when caller has CHANGEPERMISSIONS
             when beneficiary had some ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
               should pass when adding an extra allowed ERC725Y data key
               should pass when removing an allowed ERC725Y data key
               should pass when trying to clear the array completely
               should fail when setting an invalid abi-encoded array of bytes32[]
             when beneficiary had no ERC725Y data keys set under AddressPermissions:AllowedERC725YKeys:...
               should fail and not authorize to add a list of allowed ERC725Y data keys (not authorised)
               should fail when setting an invalid abi-encoded array of bytes32[]
         setting mixed keys (SETDATA, CHANGE & ADD Permissions
           when setting multiple keys
             when caller is an address with ALL PERMISSIONS
               (should pass): 2 x keys + add 2 x new permissions
               (should pass): 2 x keys + change 2 x existing permissions
               (should pass): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
             when caller is an address with permission SETDATA + ADDPERMISSIONS
               (should pass): 2 x keys + add 2 x new permissions + increment AddressPermissions[].length by +2
               (should fail): 2 x keys + add 2 x new permissions + decrement AddressPermissions[].length by -1
               (should fail): 2 x keys + change 2 x existing permissions
               (should fail): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
             when caller is an address with permission SETDATA + CHANGEPERMISSIONS
               (should pass): 2 x keys + remove 2 x addresses with permissions + decrement AddressPermissions[].length by -2
               (should pass): 2 x keys + change 2 x existing permissions
               (should fail): 2 x keys + add 2 x new permissions
               {should fail): 2 x keys + increment AddressPermissions[].length by +1
               (should fail): 2 x keys + (add 1 x new permission) + (change 1 x existing permission)
       SETDATA
         when caller is an EOA
           when setting one key
             For UP owner
               should pass
             For address that has permission SETDATA
               should pass
             For address that doesn't have permission SETDATA
               should not allow
           when setting multiple keys
             For UP owner
               (should pass): adding 5 singleton keys
               (should pass): adding 10 LSP12IssuedAssets
               (should pass): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
             For address that has permission SETDATA
               (should pass): adding 5 singleton keys
               (should pass): adding 10 LSP12IssuedAssets
               (should pass): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
             For address that doesn't have permission SETDATA
               (should fail): adding 5 singleton keys
               (should fail): adding 10 LSP12IssuedAssets
               (should fail): setup a basic Universal Profile (`LSP3Profile`, `LSP12IssuedAssets[]` and `LSP1UniversalReceiverDelegate`)
         when caller is a contract
           > contract calls
              should allow to set a key hardcoded inside a function of the calling contract
              Should allow to set a key computed inside a function of the calling contract
              Should allow to set a key computed from parameters given to a function of the calling contract
           > Low-level calls
              Should allow to `setHardcodedKeyRawCall` on UP
              Should allow to `setComputedKeyRawCall` on UP
              Should allow to `setComputedKeyFromParamsRawCall` on UP
         when caller is another UniversalProfile (with a KeyManager attached as owner)
            Alice should have ALL PERMISSIONS in her UP
            Bob should have ALL PERMISSIONS in his UP
            Alice's UP should have permission SETDATA on Bob's UP
gas used:  109258
              Alice's UP should be able to `setData(...)` on Bob's UP
           when caller has SUPER_SETDATA + some allowed ERC725YKeys
             when trying to set a disallowed key
               should be allowed to set a disallowed key: 0xce4b625ea0e12f6f6c6897ae6135917a8a1ab2f3d3e8bb1b799d3629495d4a52
               should be allowed to set a disallowed key: 0x30520d24ed35d205e2401d763abf84056b914487855139e6b5a38a3eb53a5f72
               should be allowed to set a disallowed key: 0xd725eaad24b642ab5ec92f479215e2c326d0fb18fc80ac1413070b4e913ea897
               should be allowed to set a disallowed key: 0xded639d04e7d7a12082e6ceae9b28efca1539ebc0cd3f630e0d039281df45fc3
               should be allowed to set a disallowed key: 0x9f0ee8fd4a734ff11929291ce258b12e5c2fc09653010ebac39dbf114ee5de73
             when trying to set an allowed key
               should be allowed to set the 1st allowed key
               should be allowed to set the 2nd allowed key
               should be allowed to set the 3rd allowed key
       CALL
         when interacting via `execute(...)`
           when caller has ALL PERMISSIONS
              should pass and change state at the target contract
           when caller has permission CALL
              should pass and change state at the target contract
           when caller does not have permission CALL
              should revert
           when calling a function that returns some value
              should return the value to the Key Manager <- UP <- targetContract.getName()
              Should return the value to the Key Manager <- UP <- targetContract.getNumber()
           when calling a function that reverts
              should revert
         when interacting via `executeRelayCall(...)`
           when signer has ALL PERMISSIONS
              should execute successfully
           when signer has permission CALL
              should execute successfully
           when signer does not have permission CALL
              should fail
       STATICCALL
         when caller has ALL PERMISSIONS
            should pass and return data
         when caller has permission STATICCALL
            should pass and return data
            should revert when trying to change state at the target contract
            should revert when caller try to make a CALL
         when caller does not have permission STATICCALL
            should revert
         when caller has permission STATICCALL + 2 x allowed addresses
            should revert when trying to interact with a non-allowed address
           when interacting with 1st allowed contract
              should allow to call view function -> getName()
              should allow to call view function -> getNumber()
              should revert when calling state changing function -> setName(string)
              should revert when calling state changing function -> setNumber(uint256)
           when interacting with 2nd allowed contract
              should allow to interact with 2nd allowed contract - getName()
              should allow to interact with 2nd allowed contract - getNumber()
              should revert when calling state changing function -> setName(string)
              should revert when calling state changing function -> setNumber(uint256)
         when caller has permission SUPER_STATICCALL + 2 allowed addresses
           it should bypass allowed addresses check + allow to interact with any contract
              e.g: Target Contract nb 1
              e.g: Target Contract nb 2
              e.g: Target Contract nb 3
              e.g: Target Contract nb 4
              e.g: Target Contract nb 5
       DELEGATECALL
         when trying to make a DELEGATECALL via UP, DELEGATECALL is disallowed
            should revert even if caller has ALL PERMISSIONS
            should revert even if caller has permission DELEGATECALL
            should revert with operation disallowed, even if caller does not have permission DELEGATECALL
         when caller has permission SUPER_DELEGATECALL + 2 x allowed addresses
           when calling a disallowed contract
             it should revert since DELEGATECALL is disallowed
                delegate call to contract nb 0
                delegate call to contract nb 1
                delegate call to contract nb 2
                delegate call to contract nb 3
                delegate call to contract nb 4
```

```
    when calling an allowed contract
      should revert with DELEGATECALL disallowed when trying to interact with the 1st allowed contract
      should revert with DELEGATECALL disallowed when trying to interact with the 2nd allowed contract
  DEPLOY
    when caller has ALL PERMISSIONS
     should be allowed to deploy a contract TargetContract via CREATE
     should be allowed to deploy a contract TargetContract via CREATE2
    when caller is an address with permission DEPLOY
     should be allowed to deploy a contract TargetContract via CREATE
     should be allowed to deploy a contract TargetContract via CREATE2
    when caller is an address that does not have the permission DEPLOY
      when calling via execute(...)
        should revert when trying to deploy a contract via CREATE
        should revert when trying to deploy a contract via CREATE2
      when calling via executeRelayCall(...)
        should revert when trying to deploy a contract via CREATE
        should revert when trying to deploy a contract via CREATE2
  TRANSFERVALUE
    when caller = EOA
      when recipient = EOA
        when transferring value without bytes `_data`
          should pass when caller has ALL PERMISSIONS
          should pass when caller has permission TRANSFERVALUE only
          should pass when caller has permission TRANSFERVALUE + CALL
          should fail when caller does not have permission TRANSFERVALUE
        when transferring value with bytes `_data`
          should pass when caller has ALL PERMISSIONS
          should pass when caller has permission TRANSFERVALUE + CALL
          should fail when caller has permission TRANSFERVALUE only
          should fail when caller does not have permission TRANSFERVALUE
    when caller = contract
      > Contract calls
        Should send 1 LYX to an address hardcoded in Executor (`sendOneLyxHardcoded`)
        Should send 1 LYX to an address provided to Executor (`sendOneLyxToRecipient`)
      > Low-level calls
        Should send 1 LYX to an address hardcoded in Executor (`sendOneLyxHardcodedRawCall`)
        Should send 1 LYX to an address provided to Executor (`sendOneLyxToRecipientRawCall`)
  when caller is another UP (with a KeyManager as owner)
    Alice should have ALL PERMISSIONS in her UP
    Bob should have ALL PERMISSIONS in his UP
    Alice's UP should have permission TRANSFERVALUE on Bob's UP
    Alice should be able to send 5 LYX from Bob's UP to her UP
  when caller has SUPER_TRANSFERVALUE + CALL
    should not be allowed to interact with a disallowed LSP7 contract
    should be allowed to interact with an allowed LSP7 contract
    should be allowed to interact with an allowed contract
    should be allowed to interact with an allowed contract + send some LYX while calling the function
    should be allowed to send LYX to any EOA
      should send LYX to EOA -> 0x9b31Fc7eE8bEC568A2150ba600C985C931dfEcd8
      should send LYX to EOA -> 0xC801cdABe8cF903a72cB698d1005105B0E8717BC
      should send LYX to EOA -> 0x70eb002Ba3e3804fB1A9E0Bc4b73f2b24552CCB7
      should send LYX to EOA -> 0x80f0105E987b49274009b807eff65ebcCEB209d4
      should send LYX to EOA -> 0x37665E718Aaa782177AA29358130a584F5D3B74F
    should be allowed to send LYX to any other UP contract
      should send LYX to UP 0
      should send LYX to UP 1
      should send LYX to UP 2
      should send LYX to UP 3
      should send LYX to UP 4
  when caller has TRANSFERVALUE + SUPER_CALL
    should not be allowed to do a plain LYX transfer to a non-allowed address
    should be allowed to do a plain LYX transfer to an allowed address
    should be allowed to interact with any contract
      eg: any TargetContract
        TargetContract nb 1
        TargetContract nb 2
        TargetContract nb 3
        TargetContract nb 4
        TargetContract nb 5
      eg: any LSP7 Token owned by the UP
        LSP7DigitalAsset nb 1
        LSP7DigitalAsset nb 2
        LSP7DigitalAsset nb 3
        LSP7DigitalAsset nb 4
        LSP7DigitalAsset nb 5
    should not be allowed to interact with any contract if sending LYX along the call
      Target Payable Contract nb 1
      Target Payable Contract nb 2
      Target Payable Contract nb 3
      Target Payable Contract nb 4
      Target Payable Contract nb 5
    when caller has SUPER_TRANSFERVALUE + SUPER_CALL
      should be allowed to send LYX to any address
        should send LYX to EOA -> 0x8e224dF07DA3a688EA54033c9666b153743740Da
        should send LYX to EOA -> 0x69961FB61cF5eC3B52a82320A55d73cAd051D34f
        should send LYX to EOA -> 0xd16a7848d13348962177805F4D34D371862c6c01
        should send LYX to EOA -> 0x0e440C17E534EDe83964B4Cd2B0aBE5E3D3e8D99
        should send LYX to EOA -> 0x853B1499DfBD1fb605a9b2aC69d828F1541258ce
      should be allowed to interact with any contract
        eg: any TargetContract
          TargetContract nb 1
          TargetContract nb 2
          TargetContract nb 3
          TargetContract nb 4
          TargetContract nb 5
        eg: any LSP7 Token owned by the UP
          LSP7DigitalAsset nb 1
          LSP7DigitalAsset nb 2
          LSP7DigitalAsset nb 3
          LSP7DigitalAsset nb 4
          LSP7DigitalAsset nb 5
  SIGN (ERC1271)
    can verify signature from address with ALL PERMISSIONS on KeyManager
    can verify signature from signer on KeyManager
    should fail when verifying signature from address with no SIGN permission
    should fail when verifying signature from address with no permissions set
  ALLOWEDADDRESSES
    when caller has no ALLOWED ADDRESSES set
      it should be allowed to interact with any address
        sending 1 LYX to EOA 0xd34c7cee59094ae35fe8cc0706fea9b5b8037897
        sending 1 LYX to EOA 0x29b5133aa8113bd2eda26a074f7402cd89139713
        sending 1 LYX to EOA 0xdb683d8eb4d5a6d55577a45035c8897de5dc39b0
        sending 1 LYX to EOA 0xbe4e735ea046c104b7ee567c653318625f63dbb4
        sending 1 LYX to EOA 0x06f35237e6a032e966462a97edfceba18cff64e3
    when caller has 2 x ALLOWED ADDRESSES set
      should be allowed to send LYX to an allowed address (= EOA)
      should be allowed to interact with an allowed address (= contract)
      should revert when sending LYX to a non-allowed address (= EOA)
      should revert when interacting with an non-allowed address (= contract)
    when caller has an invalid abi-encoded array set for ALLOWED ADDRESSES
      it should be allowed to interact with any address
        sending 1 LYX to EOA 0xb9411ff3f5de96cdf50ddd8cdcb7363eb2fb5fe5
        sending 1 LYX to EOA 0x9c57fd474f17546ef3624b1ce8b7747ec8d61c90
        sending 1 LYX to EOA 0x77b1134ef235ce20a63c5f070d879d94190e86fb
        sending 1 LYX to EOA 0xe1e2ee23e3e5d30b5a4577bbb5f7c179dc1bbb97
        sending 1 LYX to EOA 0x090e67f69a480a6fa56c56fd738e59af213ab431
  ALLOWEDFUNCTIONS
    when interacting via `execute(...)`
      when caller has nothing listed under AllowedFunctions
        when calling a contract
          should pass when calling any function (eg: `setName(...)`)
          should pass when calling any function (eg: `setNumber(...)`)
      when caller has 1 x bytes4 function selector listed under AllowedFunctions
        should revert when passing a random bytes payload with a random function selector
        when calling a contract
          should pass when the bytes4 selector of the function called is listed in its AllowedFunctions
          should revert when the bytes4 selector of the function called is NOT listed in its AllowedFunctions
    when interacting via `executeRelayCall(...)`
      when signer has 1 x bytes4 function selector listed under AllowedFunctions
        when calling a contract
          `setName(...)` - should pass when the bytes4 selector of the function called is listed in its AllowedFunctions
          `setNumber(...)` - should revert when the bytes4 selector of the function called is NOT listed in its AllowedFunctions
  ALLOWEDSTANDARDS
    when caller has no value set for ALLOWEDSTANDARDS (= all interfaces whitelisted)
     should allow to interact with contract that does not implement any interface
    should allow to interact with a contract that implement (+ register) any interface
      ERC1271
      LSP0 (ERC725Account)
    when caller has only ERC1271 interface ID set for ALLOWED STANDARDS
      when interacting with a contract that implements + register ERC1271 interface
```

```
            should pass
        when trying to interact an ERC725Account (LSP0)
            should allow to transfer LYX
        when interacting with contract that does not implement ERC1271
            should fail
      when caller has only LSP7 interface ID set for ALLOWED STANDARDS
        when interacting with a contract that implements + register ERC1271 interface
            should fail
        when interacting with an ERC725Account (LSP0)
            should fail when trying to transfer LYX
    ALLOWEDERC725YKeys
      keyType: Singleton
        verify allowed ERC725Y keys set
            `controllerCanSetOneKey` should have 1 x key in its list of allowed keys
            `controllerCanSetManyKeys` should have 3 x keys in its list of allowed keys
            `controllerCanSetOneKey` should have the right keys set in its list of allowed keys
            `controllerCanSetManyKeys` should have the right keys set in its list of allowed keys
        when address can set only one key
          when setting one key
            should pass when setting the right key
            should fail when setting the wrong key
          when setting multiple keys
            should fail when the list contains none of the allowed keys
            should fail, even if the list contains the allowed key
        when address can set multiple keys
          should pass when the input is all the allowed keys
          should fail when the input contains none of the allowed keys
          when setting one key
            should pass when trying to set the 1st allowed key
            should pass when trying to set the 2nd allowed key
            should pass when trying to set the 3rd allowed key
            should fail when setting a not-allowed Singleton key
          when setting 2 x keys
            should pass when
              the input is the first two (subset) allowed keys
              the input is the last two (subset) allowed keys
              the input is the first + last (subset) allowed keys
        when setting 3 x keys
          should fail when
            1st key in input = 1st allowed key. Other 2 keys = not allowed
            2nd key in input = 1st allowed key. Other 2 keys = not allowed
            3rd key in input = 1st allowed key. Other 2 keys = not allowed
            1st key in input = 2nd allowed key. Other 2 keys = not allowed
            2nd key in input = 2nd allowed key. Other 2 keys = not allowed
            3rd key in input = 2nd allowed key. Other 2 keys = not allowed
            1st key in input = 3rd allowed key. Other 2 keys = not allowed
            2nd key in input = 3rd allowed key. Other 2 keys = not allowed
            3rd key in input = 3rd allowed key. Other 2 keys = not allowed
            1st key in input = not allowed key. Other 2 keys = allowed
            2nd key in input = not allowed key. Other 2 keys = allowed
            3rd key in input = not allowed key. Other 2 keys = allowed
        when setting multiple keys
          when input is bigger than the number of allowed keys
            should fail when
              input = all the allowed keys + 1 x not-allowed key
              input = all the allowed keys + 5 x not-allowed key
            should pass when
              input contains all the allowed keys as DUPLICATE
      when address can set any key
        when setting one key
          should pass when setting any random key
        when setting multiple keys
          should pass when setting any multiple keys
    keyType: Mapping
      when address can set Mapping keys starting with a 'SupportedStandards:...'
        when setting one key
          should pass when setting SupportedStandards:LSPX
          should pass when overriding SupportedStandards:LSPX
          should pass when setting SupportedStandards:LSPY
          should pass when setting SupportedStandards:LSPZ
          should fail when setting any other not-allowed Mapping key
        when setting multiple keys
          (2 x keys) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
          (2 x keys) (override) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
          (3 x keys) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
          (3 x keys) (override) should pass when all the keys in the list start with bytes16(keccak256("SupportedStandards"))
          should fail when the list contains none of the allowed Mapping keys
          should fail, even if the list contains some keys starting with `SupportedStandards`
        when address can set any key
          when setting one key
            should pass when setting any random Mapping key
          when setting multiple keys
            should pass when setting any random set of Mapping keys
      keyType: Array
        when address can set Array element in 'MyArray[]
          when setting one key
            should pass when setting array key length MyArray[]
            should pass when setting 1st array element MyArray[0]
            should pass when setting 2nd array element MyArray[1]
            should pass when setting 3rd array element MyArray[3]
            should fail when setting elements of a not-allowed Array (eg: LSP5ReceivedAssets)
          when setting multiple keys
test gas cost:  88357
            should pass when all the keys in the list are from the allowed array MyArray[]
            should fail when the list contains elements keys of a non-allowed Array (RandomArray[])
            should fail, even if the list contains a mix of allowed + not-allowed array element keys (MyArray[] + RandomArray[])
      bytes32(0) (= zero key) as one of the allowed ERC725Y data key
          should pass when setting a data key listed in the allowed ERC725Y data keys: 0x7da57cc766f1ea8499381d54a05adf511288c2b3d84a1fee58d723eb84772f
70
          should pass when setting a data key listed in the allowed ERC725Y data keys: 0x201bc59d8e67d071c9403301d56347067ad8f4de28b0bf41dcfc265d1662bc
d4
          should pass when setting a data key listed in the allowed ERC725Y data keys: 0x0000000000000000000000000000000000000000000000000000000000000000
00
          should pass when trying to set any random data key (e.g: 0x07f83b904e398f70e60c2275e5e7b4372d1ebfaad436d5d23ff03c195d29c5ed)
          should pass when trying to set any random data key (e.g: 0xb30719f9be7ce927acc48128e57983c34807c6a0c2ec320fa0e485708b57a16d)
          should pass when trying to set any random data key (e.g: 0xe08af8ae29b8c0392e085d8cb170b52c14487abed75cc1107c0fc031feb856a7)
          should pass when trying to set any random data key (e.g: 0xe6d0d73826694058b1dab02bab34777916e23570276506b5d29c58408f366d40)
          should pass when trying to set any random data key (e.g: 0xe04c8e802ce454c81493e24a1e037b26b1de4fd24ab619cd3bb6d51bf34825e8)
      one single byte as an allowed data key (e.g: 0xaa0000...0000
        should pass when setting a data key that starts with `0xaa`
          e.g: 0xaabbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
          e.g: 0xaaccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          e.g: 0xaaddddddddddddddddddddddddddddddddddddddddddddddddddddddddddddd
        when trying to set a data key that does not start with `0xaa`
          should revert (e.g: 0xbb0000000000000000000000000000000000000000000000000000000000000000)
          should revert (e.g: 0xcc0000000000000000000000000000000000000000000000000000000000000000)
          should revert (e.g: 0xdd0000000000000000000000000000000000000000000000000000000000000000)
          should revert (e.g: 0xbb1234567800000000000000000000000000000000000000000000000000000000)
          should revert (e.g: 0xcc1234567800000000000000000000000000000000000000000000000000000000)
          should revert (e.g: 0xdd1234567800000000000000000000000000000000000000000000000000000000)
  Multi Channel nonces
    testing sequential nonces (channel = 0)
      First call > nonce should increment from undefined to NaN
      Second call > nonce should increment from undefined to NaN
      Third call > nonce should increment from undefined to NaN
      Fourth call > nonce should increment from undefined to NaN
    out of order execution (channel = n)
      channel 1
        First call > nonce should increment from 0 to 1
        Second call > nonce should increment from 1 to 2
      channel 2
        First call > nonce should increment from 0 to 1
        Second call > nonce should increment from 1 to 2
      channel 3
        First call > nonce should increment from 0 to 1
        Second call > nonce should increment from 1 to 2
      channel 15
        First call > nonce should increment from 0 to 1
  miscellaneous
    payload
      - should fail when sending an empty payload to `keyManager.execute('0x')`
      Should revert because calling an unexisting function in ERC725
    wrong operation type
      Should revert because of wrong operation type when caller has ALL PERMISSIONS
      Should revert because of wrong operation type when caller has not ALL PERMISSIONS
  Security
     Should revert when caller has no permissions set
    should revert when admin with ALL PERMISSIONS try to call `renounceOwnership(...)`
      via `execute(...)`
      via `executeRelayCall()`
```

```
        when sending LYX to a contract
          Permissions should prevent ReEntrancy and stop malicious contract with a re-entrant fallback() function.
         when calling via `executeRelayCall(...)`
           Replay Attack should fail because of invalid nonce

и-----------------------------------------------------------------|---------------------------|--------------|-----------------------------и
|                      Solc version: 0.8.10                       и  Optimizer enabled: true  и  Runs: 1000  и  Block limit: 30000000 gas
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииииииииииииииииии|ииииииииииииии|иииииииииииииииииииииииииииии
|  Methods
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  Contract               и  Method                                  и  Min          и  Max          и  Avg          и  # calls      и  usd (avg)
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  ERC725                 и  setData(bytes32[],bytes[])              и      78979  и     497621  и     266389  и          313  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  ERC725                 и  transferOwnership(address)              и      48789  и      48813  и      48812  и          313  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP0ERC725AccountInit  и  initialize(address)                     и      48845  и      74535  и      61701  и          636  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP6KeyManager         и  execute(bytes)                          и      41746  и     384843  и      68542  и         1121  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP6KeyManager         и  executeRelayCall(bytes,uint256,bytes)  и      83411  и     116201  и      95759  и           30  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP7Mintable           и  mint(address,uint256,bool,bytes)        и      86270  и      89137  и      87708  и           48  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  UniversalProfile       и  setData(bytes32[],bytes[])              и      76263  и     661191  и     264979  и          374  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  UniversalProfile       и  transferOwnership(address)              и      46139  и      46163  и      46162  и          374  и          -
ииииииииииииииииииииииииии|иииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  Deployments                                                       и                                          и  % of limit   и
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP6KeyManager                                                    и    2446081  и    2446105  и    2446104  и        8.2 %  и          -
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP6KeyManagerInit                                                и          -  и          -  и    2593136  и        8.6 %  и          -
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  LSP7Mintable                                                      и    1641836  и    1641920  и    1641838  и        5.5 %  и          -
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  UniversalProfile                                                  и          -  и          -  и    2091006  и          7 %  и          -
иииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииииии|иииииииииииии|иииииииииииии|иииииииииииии|иииииииииииииии|иииииииииииии
|  UniversalProfileInit                                              и          -  и          -  и    2202316  и        7.3 %  и          -
и-----------------------------------------------------------------|-------------|-------------|-------------|---------------|-------------и

  788 passing (9m)
  2 pending
```

Code Coverage

For code coverage the current test code managed to achieve 88.1% coverage for `ERC725Alliance/ERC725` and 82.23% coverage for `lukso-network/lsp-smart-contracts`. We recommend increasing the branch coverage to 90% before the project go live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase.

==========ERC725Alliance/ERC725:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	93.85	96.67	85.71	93.65	
ERC725.sol	0	100	0	0	24,41
ERC725Init.sol	100	100	100	100	
ERC725InitAbstract.sol	0	100	0	0	21,38
ERC725X.sol	100	100	100	100	
ERC725XCore.sol	100	96.43	100	100	
ERC725XInit.sol	100	100	100	100	
ERC725XInitAbstract.sol	100	100	100	100	
ERC725Y.sol	100	100	100	100	
ERC725YCore.sol	100	100	100	100	
ERC725YInit.sol	100	100	100	100	
ERC725YInitAbstract.sol	100	100	100	100	
constants.sol	100	100	100	100	
contracts/custom/	100	75	100	100	
OwnableUnset.sol	100	75	100	100	
contracts/helpers/	64.52	50	67.57	68.57	
Contract.sol	0	100	0	0	8
Counter.sol	100	100	100	100	
CustomRevertTest.sol	100	100	100	100	
DelegateTest.sol	0	100	0	0	14,18
ERC165InterfaceIDs.sol	100	50	100	100	
ERC725XPayableTester.sol	100	100	0	100	
ERC725YReader.sol	100	100	100	100	
ERC725YWriter.sol	100	100	100	100	
NoReceive.sol	**100**	**100**	**0**	**100**	
NonPayableFallbackContract.sol	100	100	0	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
PayableFallbackContract.sol	100	100	100	100	
Reader.sol	0	100	0	0	10,18,19,21,22
ReceiveTester.sol	100	100	100	100	
Return.sol	40	100	50	50	33,57,65
WithConstructorPayable.sol	100	100	100	100	
WithConstructorWithArgs.sol	100	100	100	100	
WithConstructorWithoutArgs.sol	100	100	100	100	
WithoutConstructor.sol	100	100	0	100	
selfdestruct.sol	100	100	100	100	
contracts/interfaces/	100	100	100	100	
IERC725X.sol	100	100	100	100	
IERC725Y.sol	100	100	100	100	
All files	**85.98**	**88.1**	**77.78**	**86.49**	

==========lukso-network/lsp-smart-contracts:

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
UniversalProfile.sol	100	100	100	100	
UniversalProfileInit.sol	100	100	100	100	
UniversalProfileInitAbstract.sol	100	100	100	100	
contracts/Custom/	50	57.14	45.45	50	
ClaimOwnership.sol	83.33	100	66.67	84.62	39,43
ERC165Checker.sol	27.78	25	20	23.53	… 6,97,98,103
IClaimOwnership.sol	100	100	100	100	
contracts/Factories/	70.97	68.75	66.67	69.7	
Create2Factory.sol	0	0	0	0	… 59,67,79,80
UniversalFactory.sol	100	91.67	100	100	
contracts/Helpers/	100	50	97.62	100	
ERC165CheckerCustomTest.sol	100	100	100	100	
ERC165Interfaces.sol	100	50	100	100	
Executor.sol	100	100	100	100	
FallbackContract.sol	100	100	100	100	
ImplementationTester.sol	100	100	100	100	
LSP2UtilsLibraryTester.sol	100	100	100	100	
NFTStorage.sol	100	100	100	100	
PayableContract.sol	100	100	66.67	100	
SignatureValidatorContract.sol	100	100	100	100	
TargetContract.sol	100	100	100	100	
contracts/Helpers/KeyManager/	81.82	100	84.62	81.82	
ERC725YDelegateCall.sol	0	100	50	0	11
KeyManagerInternalsTester.sol	88.89	100	90	88.89	41
TargetPayableContract.sol	100	100	100	100	
contracts/Helpers/Security/	100	100	100	100	
Reentrancy.sol	100	100	100	100	
contracts/Helpers/Tokens/	97.78	100	95.83	97.78	
IERC223.sol	**100**	**100**	**100**	**100**	
LSP4CompatibilityTester.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
LSP7CappedSupplyInitTester.sol	100	100	100	100	
LSP7CappedSupplyTester.sol	100	100	100	100	
LSP7CompatibleERC20InitTester.sol	100	100	100	100	
LSP7CompatibleERC20Tester.sol	100	100	100	100	
LSP7InitTester.sol	50	100	50	50	23
LSP7Tester.sol	100	100	100	100	
LSP8CappedSupplyInitTester.sol	100	100	100	100	
LSP8CappedSupplyTester.sol	100	100	100	100	
LSP8CompatibleERC721Tester.sol	100	100	100	100	
LSP8CompatibleERC721TesterInit.sol	100	100	100	100	
LSP8EnumerableInitTester.sol	100	100	100	100	
LSP8EnumerableTester.sol	100	100	100	100	
LSP8InitTester.sol	100	100	100	100	
LSP8Tester.sol	100	100	100	100	
TokenReceiverWithLSP1.sol	100	100	66.67	100	
TokenReceiverWithoutLSP1.sol	100	100	100	100	
contracts/Helpers/UniversalReceivers/	94.12	66.67	85.71	94.12	
UniversalReceiverDelegateTokenReentrant.sol	100	75	100	100	
UniversalReceiverDelegateVaultSetter.sol	100	100	100	100	
UniversalReceiverTester.sol	75	50	66.67	75	13
contracts/LSP0ERC725Account/	84.21	75	80	83.33	
LSP0Constants.sol	100	100	100	100	
LSP0ERC725Account.sol	100	100	100	100	
LSP0ERC725AccountCore.sol	93.33	75	100	92.86	124
LSP0ERC725AccountInit.sol	0	100	0	0	17,25
LSP0ERC725AccountInitAbstract.sol	100	100	100	100	
contracts/LSP10ReceivedVaults/	100	100	100	100	
LSP10Constants.sol	100	100	100	100	
contracts/LSP1UniversalReceiver/	100	100	100	100	
ILSP1UniversalReceiver.sol	100	100	100	100	
ILSP1UniversalReceiverDelegate.sol	100	100	100	100	
LSP1Constants.sol	100	100	100	100	
LSP1Utils.sol	100	100	100	100	
contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/	100	50	100	100	
LSP1UniversalReceiverDelegateUP.sol	100	50	100	100	
contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/Handling/	85	78.57	100	100	
TokenAndVaultHandling.sol	85	78.57	100	100	
contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/	100	50	100	100	
LSP1UniversalReceiverDelegateVault.sol	100	50	100	100	
contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/Handling/	88.24	80	100	100	
TokenHandling.sol	88.24	80	100	100	
contracts/LSP2ERC725YJSONSchema/	63.33	81.25	50	61.11	
LSP2Errors.sol	100	100	100	100	
LSP2Utils.sol	**63.33**	**81.25**	**50**	**61.11**	**… 196,197,199**
contracts/LSP3UniversalProfile/	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
LSP3Constants.sol	100	100	100	100	
contracts/LSP4DigitalAssetMetadata/	100	100	100	100	
ILSP4Compatibility.sol	100	100	100	100	
LSP4Compatibility.sol	100	100	100	100	
LSP4Constants.sol	100	100	100	100	
LSP4DigitalAssetMetadata.sol	100	100	100	100	
LSP4DigitalAssetMetadataInitAbstract.sol	100	100	100	100	
LSP4Errors.sol	100	100	100	100	
contracts/LSP5ReceivedAssets/	100	83.33	100	98.21	
LSP5Constants.sol	100	100	100	100	
LSP5Utils.sol	100	83.33	100	98.21	92
contracts/LSP6KeyManager/	96.59	90.91	100	99.42	
ILSP6KeyManager.sol	100	100	100	100	
LSP6Constants.sol	100	100	100	100	
LSP6Errors.sol	100	100	100	100	
LSP6KeyManager.sol	100	50	100	100	
LSP6KeyManagerCore.sol	96.32	92.19	100	99.36	666
LSP6KeyManagerInit.sol	100	100	100	100	
LSP6KeyManagerInitAbstract.sol	100	50	100	100	
LSP6Utils.sol	100	100	100	100	
contracts/LSP7DigitalAsset/	94.44	85	100	91.57	
ILSP7DigitalAsset.sol	100	100	100	100	
LSP7Constants.sol	100	100	100	100	
LSP7DigitalAsset.sol	100	100	100	100	
LSP7DigitalAssetCore.sol	93.85	85	100	90.79	… 253,254,256
LSP7DigitalAssetInit.sol	100	100	100	100	
LSP7DigitalAssetInitAbstract.sol	100	100	100	100	
LSP7Errors.sol	100	100	100	100	
contracts/LSP7DigitalAsset/extensions/	94.29	66.67	92.31	89.47	
ILSP7CappedSupply.sol	100	100	100	100	
ILSP7CompatibleERC20.sol	100	100	100	100	
LSP7CappedSupply.sol	100	50	100	75	20
LSP7CappedSupplyCore.sol	100	100	100	100	
LSP7CappedSupplyInitAbstract.sol	100	50	100	75	19
LSP7CompatibleERC20.sol	80	100	83.33	80	65
LSP7CompatibleERC20Core.sol	100	100	100	100	
LSP7CompatibleERC20InitAbstract.sol	83.33	100	83.33	83.33	65
contracts/LSP7DigitalAsset/presets/	50	100	50	50	
ILSP7Mintable.sol	100	100	100	100	
LSP7CompatibleERC20Mintable.sol	0	100	0	0	20
LSP7CompatibleERC20MintableInit.sol	0	100	0	0	14,28
LSP7CompatibleERC20MintableInitAbstract.sol	0	100	0	0	16,25
LSP7Mintable.sol	100	100	100	100	
LSP7MintableInit.sol	100	100	100	100	
LSP7MintableInitAbstract.sol	**100**	**100**	**100**	**100**	
contracts/LSP8IdentifiableDigitalAsset/	95.56	95.65	92.86	96.23	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ILSP8IdentifiableDigitalAsset.sol	100	100	100	100	
LSP8Constants.sol	100	100	100	100	
LSP8Errors.sol	100	100	100	100	
LSP8IdentifiableDigitalAsset.sol	100	100	100	100	
LSP8IdentifiableDigitalAssetCore.sol	97.65	95.65	100	98.02	104,130
LSP8IdentifiableDigitalAssetInit.sol	0	100	0	0	19,33
LSP8IdentifiableDigitalAssetInitAbstract.sol	100	100	100	100	
contracts/LSP8IdentifiableDigitalAsset/extensions/	100	77.78	100	97.33	
ILSP8CappedSupply.sol	100	100	100	100	
ILSP8CompatibleERC721.sol	100	100	100	100	
ILSP8Enumerable.sol	100	100	100	100	
LSP8CappedSupply.sol	100	50	100	75	20
LSP8CappedSupplyCore.sol	100	100	100	100	
LSP8CappedSupplyInitAbstract.sol	100	50	100	75	21
LSP8CompatibleConstants.sol					
LSP8CompatibleConstants.sol	100	100	100	100	
LSP8CompatibleERC721.sol	100	100	100	100	
LSP8CompatibleERC721Core.sol	100	100	100	100	
LSP8CompatibleERC721InitAbstract.sol	100	100	100	100	
LSP8Enumerable.sol	100	100	100	100	
LSP8EnumerableCore.sol	100	66.67	100	100	
LSP8EnumerableInitAbstract.sol	100	100	100	100	
contracts/LSP8IdentifiableDigitalAsset/presets/	50	100	50	50	
ILSP8Mintable.sol	100	100	100	100	
LSP8CompatibleERC721Mintable.sol	0	100	0	0	20
LSP8CompatibleERC721MintableInit.sol	0	100	0	0	14,28
LSP8CompatibleERC721MintableInitAbstract.sol	0	100	0	0	16,25
LSP8Mintable.sol	100	100	100	100	
LSP8MintableInit.sol	100	100	100	100	
LSP8MintableInitAbstract.sol	100	100	100	100	
contracts/LSP9Vault/	100	75	100	100	
LSP9Constants.sol	100	100	100	100	
LSP9Vault.sol	100	100	100	100	
LSP9VaultCore.sol	100	75	100	100	
LSP9VaultInit.sol	100	100	100	100	
LSP9VaultInitAbstract.sol	100	100	100	100	
contracts/Legacy/	0	0	0	0	
UniversalReceiverAddressStore.sol	0	0	0	0	… 53,55,60,61
contracts/Legacy/Registries/	100	75	100	100	
AddressRegistry.sol	100	100	100	100	
AddressRegistryRequiresERC725.sol	100	66.67	100	100	
contracts/Utils/	100	100	100	100	
UtilsLib.sol	100	100	100	100	
All files	**90.57**	**82.23**	**88.28**	**90.79**	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
55d53041c3688418f96406ef4541bc9549bf8be15a67e8f67f6c79ce8d70f496 ./contracts/lsps/UniversalProfile.sol

34acb82e1344a8eb8ba372decce84a72d609da72122d2ee639e05fe8adbecef3 ./contracts/lsps/UniversalProfileInit.sol

176fee542581697ef8647b5c6bd0d7243a570f2a13e0c2bebcd8351033714459 ./contracts/lsps/UniversalProfileInitAbstract.sol

5b9fccdc2f06c2e7f38137ddd063e90a94d24227e0c667f3e90b89507f816d14 ./contracts/lsps/Utils/UtilsLib.sol

c92d24adf8dd80e3a08187a890fb1d416b92ec909eda71b5c575ae8ea152791f ./contracts/lsps/LSP9Vault/LSP9Constants.sol

826cb3cd08bda32a5b27627e84f9f6d5a74a58390f61478a5677da20d884a5df ./contracts/lsps/LSP9Vault/LSP9Vault.sol

b003e93de34ec7ab1dd58bc134a4d90222d63cfc1ae435f34ec38936e2fc07c5 ./contracts/lsps/LSP9Vault/LSP9VaultCore.sol

3f2f41d0bd67fe94d5b16fe3fbf0ed8da735c40c5ac61d853fa4a3c449f18f7a ./contracts/lsps/LSP9Vault/LSP9VaultInit.sol

542cf93829c374a1bab98ea14fa3b8f182a92a18841b7d34f26633fa069957a9 ./contracts/lsps/LSP9Vault/LSP9VaultInitAbstract.sol

d45d0961e2d21342d3e173684d35795111d1f401ec9fc47aec4439605606e679
./contracts/lsps/LSP8IdentifiableDigitalAsset/ILSP8IdentifiableDigitalAsset.sol

48c8861c56027721efd5452c9994bf9805d0a24dcf33ea87817ec60ec5e87b50 ./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8Constants.sol

61b55a9434d232c08d73167ba6c4bf8f78562deee33b1ad6f5de657535ecc7c8 ./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8Errors.sol

adc5af3b291c1dcca385869ca316cbc810bc4f1686171288f198c259f1e9a1f8
./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAsset.sol

086116de814fcea7206e19d59ba8a7cf939f976b86d7324770a4f27a22d692d6
./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetCore.sol

02546c7be5e2b5ab723ca3dcdd65a75f0a01d2b9cca867bfafe534409f93cf0c
./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetInit.sol

23fb0ac0645352401018fe154a010a3a93f8cd6299d1b310727b329e718712e0
./contracts/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAssetInitAbstract.sol

c846cde9cfb5a5a158b6d4ab92bccf91050d49be5c03b6f96d9da1915fe0386a ./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/ILSP8Mintable.sol

386b8b0ca4d1aec7d41b8516d7c6f3d433b0e9c2af2ed484fb9c1f087325da97
./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8CompatibleERC721Mintable.sol

1d34738a5c9e4f1db876c803bc48d075796e3d935034e19513c8fd13b0bcf648
./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8CompatibleERC721MintableInit.sol

4dbecce3d569276167c531cac131e08204f52bcdc22498571e060a1d43d41058
./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8CompatibleERC721MintableInitAbstract.sol

bf8853c23a0577ff737d9d88cf3227d23cd97166e64aa89b648f9afaf0fd0856 ./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8Mintable.sol

8d1bdef15ce86b2aaae350f7484a3e8cff632dd8d6c463fb575cf782aa4ab0fc ./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8MintableInit.sol

54ea26e8f7098b20812e6445e61400fa9fdfb0eede14e53ab5e5501d1c1d59bf
./contracts/lsps/LSP8IdentifiableDigitalAsset/presets/LSP8MintableInitAbstract.sol

57292084871549112129 19aec0b03d9b07e385575026725763ad252c813b74b2
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/ILSP8CappedSupply.sol

97571c7430612df9a2d07145eaf627420f28161e3bd0a5479b37d56c3c6d0544
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/ILSP8CompatibleERC721.sol

1e869bf90bdc5d84e74c2d45fb405c46c2bba48cf4c3b3d5d329a7b8d47f4462 ./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/ILSP8Enumerable.sol

5a4bc7805b1cae9139a42233820f40d3a1a0ee64b9e55eeacd07d5e021156046 ./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupply.sol

b61bffd32f4c45af95c6f8d81647dbf0eae37623a7ee3b1a5417e0d1ab409f82
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupplyCore.sol

f87e136cb060a2a6b35c1e4f7ddd7a5853cb6807865067f104806716b12a6a61
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupplyInitAbstract.sol

649b496636e2d36cf113c259765cb48111e2fb401e145079938f4058fc8c9c9c
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleConstants.sol

5b50295791d71d114cffaf719d514ad614f2e84264154aa9f68319267982ff0b
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721.sol

2bd9467b2c0c78d85f0aea0094fc3c4080dcffd066d416ee819d95122e0c4207
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721Core.sol

ed26a60d981293721c3dc7feab45b747b54ee6e9a02f680f76e8301f97d00239
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721InitAbstract.sol

bac342a9b570f5dfeef35ec30d920e8221cc3456b3a7c6957cd34997553f20b8 ./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8Enumerable.sol

6176723391a567ce05d4a1a60af3e5ce1d8a3b7c4c14976d1fba93dc50904382
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8EnumerableCore.sol

2fadfbeff3bb1287a77fe8a8b205e63ce29bdd4559e6213c9d230fbfd56a7c55
./contracts/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8EnumerableInitAbstract.sol

4dea698aee81ca2f1df30f2f91a4ff6ffd646fad895837ae012cbe657e74bb47 ./contracts/lsps/LSP7DigitalAsset/ILSP7DigitalAsset.sol

b2397b4456c8b4964d1dac632544dc4f4b5b3ee71506fb5a1b57a03cb15e696c ./contracts/lsps/LSP7DigitalAsset/LSP7Constants.sol

7b970e867d11037b64081be9916e3ec7aea04cac473312c042199df9ee91f7ac ./contracts/lsps/LSP7DigitalAsset/LSP7DigitalAsset.sol

5b04742938621d445a50698a1b5ab23dd5533bba98dfcb75c6467901cf2dbc25 ./contracts/lsps/LSP7DigitalAsset/LSP7DigitalAssetCore.sol

95894bd41ae7bd50b356f0c55f2445e85de17c2bfda02fd614bfcc4240fe56df ./contracts/lsps/LSP7DigitalAsset/LSP7DigitalAssetInit.sol
```

0295af381757e1a4d4d5235bf6af06736a2d81867ecb43a371335215b575af9b ./contracts/lsps/LSP7DigitalAsset/LSP7DigitalAssetInitAbstract.sol

9b7c36e9c193761cf962e55aae18b99e7ed6c2b1ef70a581603ed581cf81778d ./contracts/lsps/LSP7DigitalAsset/LSP7Errors.sol

ea0618aa6448b5b6e3def903fcc1da7a5973f9db1bbb463f5808dbc7c6305c76 ./contracts/lsps/LSP7DigitalAsset/presets/ILSP7Mintable.sol

944afa753160e7971df156590ffa2b5307e15f24e6688814c9e0d313ed711487 ./contracts/lsps/LSP7DigitalAsset/presets/LSP7CompatibleERC20Mintable.sol

09631b14f11bea9c498a7ba9031932bd5a17999ecc3887642c07481ee0450fdc ./contracts/lsps/LSP7DigitalAsset/presets/LSP7CompatibleERC20MintableInit.sol

eff45b13d7adafd71264ef1c0ff11cc68422d82b909b5879fe0393db312b040c
./contracts/lsps/LSP7DigitalAsset/presets/LSP7CompatibleERC20MintableInitAbstract.sol

86a505365be7f79b289005e24a0121b44e25db87ebe9f09a4f74874b18af7ad4 ./contracts/lsps/LSP7DigitalAsset/presets/LSP7Mintable.sol

bbc88a063584b3dac2ca4c6085f9c15c9af1e55b14dcfb3364213bd97c280948 ./contracts/lsps/LSP7DigitalAsset/presets/LSP7MintableInit.sol

2d6f6ebb18a3a42e1ea940993892b07f708d23288c5c65a01ef1a347add38f79 ./contracts/lsps/LSP7DigitalAsset/presets/LSP7MintableInitAbstract.sol

0f398b817f1aa2f92419694e66aad333b4b5af06937bc6715c62530e5636d1e0 ./contracts/lsps/LSP7DigitalAsset/extensions/ILSP7CappedSupply.sol

fbd2a3f4ee8201f704c98c249d00de3a724ff27a66613f525804a437b141d55e ./contracts/lsps/LSP7DigitalAsset/extensions/ILSP7CompatibleERC20.sol

32338e07234c7db76db5812aaa0218b7c94e99f2b66a503774662da46b792c56 ./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CappedSupply.sol

7cd4347e99b8fc77de9626d427f36572db1ed5e77cef5ead0e894568c4d7e849 ./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CappedSupplyCore.sol

d5c6d2fc4f2665befc634476a899425730ecdc86856d78e79cba801c60d32aa9 ./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CappedSupplyInitAbstract.sol

b58ed962893522bffdbdff5d621557ad2ac989e4b11d7fa0902578ea4e3d82ed ./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CompatibleERC20.sol

207aff1685af76dbe4099568b2fa061c69216971020d83423dccf2a3f33084ac ./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CompatibleERC20Core.sol

5f1bf9371f209b3bb489554224d19322ccf0c5d45aa5c9356676e2b7ab4139f5
./contracts/lsps/LSP7DigitalAsset/extensions/LSP7CompatibleERC20InitAbstract.sol

72ea4854b2aa3c6ce8269cd4f5668dc4430e62d39b8ac330de9e813c593cc922 ./contracts/lsps/LSP6KeyManager/ILSP6KeyManager.sol

0dcf6e214d96f2dc82bc0fa4b3f6d8b3b7a9d131ffe78d63134c28810a37a050 ./contracts/lsps/LSP6KeyManager/LSP6Constants.sol

425eb1ef9e8d9c281c68f35a76dcaf079643a32b5db75def31d09c15ef062cc0 ./contracts/lsps/LSP6KeyManager/LSP6Errors.sol

fc94ba2b868b00edc2907c24696c88737b4ddad120ca60ddd49d3cd705f45171 ./contracts/lsps/LSP6KeyManager/LSP6KeyManager.sol

672a75e94e30a451500af4afcac92c3596ce5a7713c5e64d2b9801a5ab7eab26 ./contracts/lsps/LSP6KeyManager/LSP6KeyManagerCore.sol

2f1945e81d8a4c46ca1d2d33243cd564fdcfc05032725e82dde81846c9f52c25 ./contracts/lsps/LSP6KeyManager/LSP6KeyManagerInit.sol

da8bbc4052756858c80d6162770a17a0c03f40f3a73688954643ad32e85b4c1c ./contracts/lsps/LSP6KeyManager/LSP6KeyManagerInitAbstract.sol

c3f22f60359ea2c24d1a0bcf49d7f2efac01cc74476dfb07206428df680a2f14 ./contracts/lsps/LSP6KeyManager/LSP6Utils.sol

f85c3244b36ae76e38a9cb62108368414bb11fade710a6a938accb47562d800d ./contracts/lsps/LSP5ReceivedAssets/LSP5Constants.sol

c72f5bfc00936a5dabc9686c7c792c7a5b1fbe5555681c0d4a2e65e6c1c460ac ./contracts/lsps/LSP5ReceivedAssets/LSP5Utils.sol

47645357e64d0dbfed6dcd999ffd2b66beb9bc0f043a591ba64fac9db905a35d ./contracts/lsps/LSP4DigitalAssetMetadata/ILSP4Compatibility.sol

964508657a9a87969ac8d8cdf6a5b62750306fc751cfc8c8ef16cb66bc964455 ./contracts/lsps/LSP4DigitalAssetMetadata/LSP4Compatibility.sol

df1493a7746eaa2c7d28fa69a42fe628adc99d79aa977a2ca502eef73e6ce04c ./contracts/lsps/LSP4DigitalAssetMetadata/LSP4Constants.sol

45734da9ed85f60659fdc967b6fff74a347bee8806b046dd7a61ef170f634e6e ./contracts/lsps/LSP4DigitalAssetMetadata/LSP4DigitalAssetMetadata.sol

07e0547b22a99f75b1690d946ccfb75ae9c4265d98d7cc5d9b9d2a4caed848b3
./contracts/lsps/LSP4DigitalAssetMetadata/LSP4DigitalAssetMetadataInitAbstract.sol

0f6500778cb5bb9918bc260463fe6d492026d4bd47d96b72f88b9a0a950081bf ./contracts/lsps/LSP4DigitalAssetMetadata/LSP4Errors.sol

08f9bc0a4a67e117d0773f07bec8dd930c0364a0ab723abec2e0c6cfa7b51cc6 ./contracts/lsps/LSP3UniversalProfile/LSP3Constants.sol

a684c6e799f8f5aaccd63136a246d197bd95c52be436fa84b42c2f1ad1e47940 ./contracts/lsps/LSP2ERC725YJSONSchema/LSP2Errors.sol

4d40f91b2aa6d601960eab07fad2b462cf113ae38b4f060a8e45824af3efd21e ./contracts/lsps/LSP2ERC725YJSONSchema/LSP2Utils.sol

832db54ceb8a63e152ffb65cbd169b7f5b63d2fd99e4ce8460aadb96f52290b0 ./contracts/lsps/LSP1UniversalReceiver/ILSP1UniversalReceiver.sol

f0c658b4b614ccf255e10cde2be231fe28ccafd3f93c17abb8b1f78351293fb8 ./contracts/lsps/LSP1UniversalReceiver/ILSP1UniversalReceiverDelegate.sol

18b480a767d26e9d5f865f3b281e7d8568392fae90e2b52fb82bf9517240c6c4 ./contracts/lsps/LSP1UniversalReceiver/LSP1Constants.sol

344523566a854db34b571e98b1afb989c613f3644da5affd620619f6f4399c2d ./contracts/lsps/LSP1UniversalReceiver/LSP1Utils.sol

c3ea3d2db58ae63d04340a10c4a0e05f6ff51e124a76c8e3cb5212c697f2541a
./contracts/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/LSP1UniversalReceiverDelegateVault.sol

944b568e584a5630c7235d91c2014cc808cfc51fd8bde36517eb29390fb00cdd
./contracts/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/Handling/TokenHandling.sol

a1b7e3e158251eeb472f6745fa0788c49c3f9b985f28925c963c89522b227841
./contracts/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/LSP1UniversalReceiverDelegateUP.sol

4ff5562c2ec9e2cab14f5c1e7f4fc2d564e2e55ce9a1efa4d19c4e0177a10982
./contracts/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP/Handling/TokenAndVaultHandling.sol

dfd8db1f9275efef3c3ab70a79db7f20253655ea4e95f6aac96a3a5fefb63661 ./contracts/lsps/LSP10ReceivedVaults/LSP10Constants.sol

43428032cf4e531a1996544c3492770750c3318a33a910380e7f15b051df0913 ./contracts/lsps/LSP0ERC725Account/LSP0Constants.sol

4737600fb721826498a196dd2ac2cd64e2c049647c1b6d94776785dfbe6ebe2c ./contracts/lsps/LSP0ERC725Account/LSP0ERC725Account.sol

e0dd1d6574737a989486ff76f9753faf453fe73237fd703dba252b8378388614 ./contracts/lsps/LSP0ERC725Account/LSP0ERC725AccountCore.sol

1355ddf00c2a2274a01b7c88f49516ff2373646dcc0f6982f67cf961df92b1de ./contracts/lsps/LSP0ERC725Account/LSP0ERC725AccountInit.sol

330718003e2f78216e50ea7b7c43df23e669c8e3b8c35d2c75293b3d29726def ./contracts/lsps/LSP0ERC725Account/LSP0ERC725AccountInitAbstract.sol

36515c652ade841e428e773eb458c5844640af88896cb9b4d4be50962ba651d9 ./contracts/lsps/Legacy/UniversalReceiverAddressStore.sol

989c7f94ac472c55cd0e9ff7260e7fa99559d8dc9168f3fb659c621a4b9599b0 ./contracts/lsps/Legacy/Registries/AddressRegistry.sol

84ba2368b288e4e66615a95c8b519108bf3a00da68f873bc8d5ada14c9b10316 ./contracts/lsps/Legacy/Registries/AddressRegistryRequiresERC725.sol

6742a7797bc37f7340d4a44f8e34bcb5573a5908f0976de07b65b05cb03264fa ./contracts/lsps/Factories/Create2Factory.sol

d7314aa55b5d5797f64bef6ace230d62d210d29e8131ec5f481af12f5d97420b ./contracts/lsps/Factories/UniversalFactory.sol

9e9b0da97a355b50d5bbc4c2f24c317bfd7204a1793ca207db1d80a8b75cada8 ./contracts/lsps/Custom/ClaimOwnership.sol

db043af168308bfaf9eccfc373c3609a98c3bb2157c74f237e7e37bf46a26746 ./contracts/lsps/Custom/ERC165Checker.sol

afbdb4915fe30fd8db64e353f060ebe13e9e47cdf055569df44a459a0e8d5f2a ./contracts/lsps/Custom/IClaimOwnership.sol

fc3bb4bbffc88acb648fe6dbd1dc4c21f38af7725bd3482e535f8106480a3dec ./contracts/erc725/constants.sol

e6f1513b7bee45696c16c297eb02c5d6c0e7862fe9308977aee4130622e5f670 ./contracts/erc725/ERC725.sol

07019830df3ad4f9d8ae56089b048b15febbc913d6e8c1be3a0dadf83242800f ./contracts/erc725/ERC725Init.sol

23e2bed0574a91e9b170d5ab456045d6fac7d1322a0703a9e049e8921b5d4497 ./contracts/erc725/ERC725InitAbstract.sol

fdf11fda029d99e835db919bb854dd0840143390d022c231f6156c8f76300321 ./contracts/erc725/ERC725X.sol

e465083820795761e93ce212c7a9184ce826ea412c8038ebe35868ae232ef993 ./contracts/erc725/ERC725XCore.sol

95f3f58ef9e33f2ce45050768cc7eff00f27cb8a84ff65fc73a568ffea3e1129 ./contracts/erc725/ERC725XInit.sol

9d87f0b754fe420629beb1b12823e76696733e46d3391757e9e5ad242c805a87 ./contracts/erc725/ERC725XInitAbstract.sol

68c77a66b03b9bbef3b18036908a25be8a6cbab12289d60066afe3b10eec5825 ./contracts/erc725/ERC725Y.sol

ea4180a6a773af7b7b5f30f7feaabc78bdc23a3d893b0a97124417025892883c ./contracts/erc725/ERC725YCore.sol

715abca6d32d2f52f43ad95071981944402d1fcff3ea6d576043daac0b154479 ./contracts/erc725/ERC725YInit.sol

e85b04661aaf88ca9fd28b2e7c3d2bafdaa893628066133d26b2462b7817670e ./contracts/erc725/ERC725YInitAbstract.sol

c427249e83b8cb5e5a83cb61dd3ac7c571dc44177fd5d002ef45944156abc957 ./contracts/erc725/interfaces/IERC725X.sol

c133130c38d17a455938c68b2a25404e1a69150c2b8d99677f7890680f31c53e ./contracts/erc725/interfaces/IERC725Y.sol

eaea6a07652b111a2002bed996565e43f25af595b008b5de2588e9ebc287f207 ./contracts/erc725/custom/OwnableUnset.sol

Tests

e59d223f899594d5e3f924570161f594cd171543317f9e517e12ea7753dd2db3 ./tests/lsps/ClaimOwnership.behaviour.ts

46c3d3ecacae270e07e77b5ffd4d1f79b4f61d04ff07944bcc1ba8dddd24f7da ./tests/lsps/UniversalProfile.behaviour.ts

414e9009c74130844023b750b3c2e890908b473ea64d10f17fb11bae5d25a825 ./tests/lsps/UniversalProfile.test.ts

2541384b08fc48327ebf9423373312a4c13df1cf866fd9c8ec07324f753cff5c ./tests/lsps/utils/context.ts

80175446f7a7cd26ef47c45d1e5be0f88adcc3ea25b8b1ab2a0a69fd46dbcc6f ./tests/lsps/utils/deploy.ts

371ca6078fb59e9e89d109a29f7c9d3a27a9db1728aed99298d5c9f8ce73b53a ./tests/lsps/utils/errors.ts

4391b35a8fd4dceb2ec2055d002f9e2f02aa951878021971b7e0ec6acccb13b1 ./tests/lsps/utils/fixtures.ts

3372a900edc08834288627c445cd83634ddbe0ffcf3a4ca78bfd99768ca64160 ./tests/lsps/utils/helpers.ts

baf7d1f167a13794f29ca35acb3e30158e3da6a1aae05a2abee6eb7dcec7a313 ./tests/lsps/utils/tokens.ts

dccbff92d26c28e15386fbd783f6a782a889c3b6a9e56a8994074ce694be5627 ./tests/lsps/LSP9Vault/LSP9Vault.behaviour.ts

d21f843a0b11fa6de466979c7f71bc24046092fe65a2f692ca1c55cdc126d1b7 ./tests/lsps/LSP9Vault/LSP9Vault.test.ts

afe590575ec035608fef9e8d76e07c4efc5c84d4e038a9c0c961944d9b4447bf
./tests/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAsset.behaviour.ts

287d8ee89ffeb8155189131a6018a6fa67a1c9d356e5e7cf0a646e96652d38d4
./tests/lsps/LSP8IdentifiableDigitalAsset/LSP8IdentifiableDigitalAsset.test.ts

4ad83deb05c6c15f0d92d9597f3590a44e5eded672989a037bdbcd2426ca4537
./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupply.behaviour.ts

25b0d3352314c9d50ca6208d63126039c6ea74078c7a32a5e7a99a88e873b798 ./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CappedSupply.test.ts

2d0930182296bf59c024b6bcc6dc8e9af375c64bcb814e325057364f2c396ad0
./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721.behaviour.ts

19b0c407e9ab4810c3dfac83dc73eb4cb472f6d9b3395b067d8f21e67c29ed45
./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8CompatibleERC721.test.ts

e8d718eb5f797f17f9c330f3e92e9cc035324f9b433f925fafdd509b70fb790e
./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8Enumerable.behaviour.ts

6542731705328ec6fa525ba42d09892d9b98185d8787988550953d4af945af06 ./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8Enumerable.test.ts

77f109aa7f64c1b41b7ec52dca46113e991777871772568f94f6dc62dfea66ba
./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8Mintable.behaviour.ts

a4f6332da68c8be339bdc4d0cc4708867f889ec1b2a715ed0f660419277799aa ./tests/lsps/LSP8IdentifiableDigitalAsset/extensions/LSP8Mintable.test.ts

875b56b44d68ee578d77b19c3b1acb1cafc24c09be246e9caf123cf6217c2e2c ./tests/lsps/LSP7DigitalAsset/LSP7DigitalAsset.behaviour.ts

a56579a00e19552cd3b0d6cd57a45050bb752716a1915343e57cdda54c880a28 ./tests/lsps/LSP7DigitalAsset/LSP7DigitalAsset.test.ts

e3eade8d3b20fef5ea420897578f8a672d16ce9254acc85a6c11e6b0d9b0b130 ./tests/lsps/LSP7DigitalAsset/extensions/LSP7CappedSupply.behaviour.ts

41cca74924091be0a17f5afdb18945859b6a1e740c7cc07b61365b9111ad95dc ./tests/lsps/LSP7DigitalAsset/extensions/LSP7CappedSupply.test.ts

d620c836e69c63150e4b439f15d5fdc9fc8866062a07cf66afb06758b8ebd2e1 ./tests/lsps/LSP7DigitalAsset/extensions/LSP7CompatibleERC20.behaviour.ts

e363004a8a1fe8551200f3875a9683cf4e75fa4a07b919da249b4b1cdfe61f64 ./tests/lsps/LSP7DigitalAsset/extensions/LSP7CompatibleERC20.test.ts

6540d40469b51cc0b42a78a38d2d3a956f31fdb45619d96fa2b359f8d4cffb6a ./tests/lsps/LSP7DigitalAsset/extensions/LSP7Mintable.behaviour.ts

4b0a9403df5c579805e9b39a284e6aa9dd3636ede14d72c7313dc9bca1b58ce3 ./tests/lsps/LSP7DigitalAsset/extensions/LSP7Mintable.test.ts

419ce2d1046d2059af30fc93945ee26654403204fab3e8f0a08cf6bcdfdd8d93 ./tests/lsps/LSP6KeyManager/LSP6KeyManager.behaviour.ts

ac286c9508d4e2921b7875f95d0731feb170d560a27654eea7a9b51364ac9679 ./tests/lsps/LSP6KeyManager/LSP6KeyManager.test.ts

81b9a195b1b7019f0bfd81fe26a490f1bd6bf56e171c5dde30b6bd7ecb5bb307 ./tests/lsps/LSP6KeyManager/tests/AllowedAddresses.test.ts

d654af1c5b4e3161ba87466cc3fd73342942597cea5940b8d57b5035d2122c0d ./tests/lsps/LSP6KeyManager/tests/AllowedERC725YKeys.test.ts

b778648bfe8f30f36b561dabb682a174737b41e74f4fa857cb34c9ae7cdbd448 ./tests/lsps/LSP6KeyManager/tests/AllowedFunctions.test.ts

b96ef6a4ad7e1ffaa7260cef865fce26dfc23973530b14685f0667dc7aa818ec ./tests/lsps/LSP6KeyManager/tests/AllowedStandards.test.ts

947847e4bde5f7c06b226685257a7d5b1c52a4fbcf5c4a054093943f64882e30 ./tests/lsps/LSP6KeyManager/tests/index.ts

0702c5d640ee19a4f76b86de1d3ffb10806c5fbd4934d6c6b55427130b6358e0 ./tests/lsps/LSP6KeyManager/tests/MultiChannelNonce.test.ts

a289bb0562c032cf7692b66c6a63ee2442e885e3675f5be3a420bfb6960ed5b4 ./tests/lsps/LSP6KeyManager/tests/OtherScenarios.test.ts

97e16446b8a2daec513d54136c79ce001d67649b90ec513077fd86c34f2bde29 ./tests/lsps/LSP6KeyManager/tests/PermissionCall.test.ts

c7e58cabe37d529ac5f186c59677f9d43b6cf6194f89a26df9f7858c201d18a9 ./tests/lsps/LSP6KeyManager/tests/PermissionChangeAddPermissions.test.ts

ef7a5c77e2bc13246fa21daa7f26b52a2322f4541eab8df25f6040ddaa99c09c ./tests/lsps/LSP6KeyManager/tests/PermissionChangeOwner.test.ts

11f125c4025e6bfb973c08af3454952a5783b1a96cdc4136e101afb70ee4f5f0 ./tests/lsps/LSP6KeyManager/tests/PermissionDelegateCall.test.ts

2b730b6d332add17809085e459b3d87e77c7102ab928a9767f190f469f2bc04b ./tests/lsps/LSP6KeyManager/tests/PermissionDeploy.test.ts

9e38d5376c213544c414abd0f7e5cf6128b857ad06bad2733d8dee18b82351a5 ./tests/lsps/LSP6KeyManager/tests/PermissionSetData.test.ts

4355ee57e566eeb1b189affae48d16f77a296fb5a41a13a049afbc289906d072 ./tests/lsps/LSP6KeyManager/tests/PermissionSign.test.ts

e54dbe48767e133b9002174c5f301024487e4ce6e19d0da57e4f5fa26338fbe6 ./tests/lsps/LSP6KeyManager/tests/PermissionStaticCall.test.ts

3028bbfea3df0bc520e9a975f94c59b6d744a64a5b87ec4e7eaeedf5ba3cc55c ./tests/lsps/LSP6KeyManager/tests/PermissionTransferValue.test.ts

673466dcf409138fa4312adba933b1caf05556d0f12d86ff23fd6b10cdac5e92 ./tests/lsps/LSP6KeyManager/tests/Security.test.ts

6cb2b105258ec8328bff39bf41764a79491337f94d561b3078340a15830b35cc ./tests/lsps/LSP6KeyManager/internals/AllowedAddresses.internal.ts

4ccf2efd9f0c3b13c1255447edbef926d2e4473274ccda70a0049310a6f22044 ./tests/lsps/LSP6KeyManager/internals/AllowedERC725YKeys.internal.ts

d8fc02e2c11cbf3c8001facb655bdf8e25c04934860ef2e8f13918a4cd108b1e ./tests/lsps/LSP6KeyManager/internals/AllowedFunctions.internal.ts

8434e2fccc1af860e2a8588eace5bab67f1e2335e23d28da4df6c86dc58ab537 ./tests/lsps/LSP6KeyManager/internals/index.ts

964b80755d94e81e10cb37df0cf4f28588d829ec580ce9a241bf15f5f7f73356 ./tests/lsps/LSP6KeyManager/internals/ReadPermissions.internal.ts

16293162529c7447f1436a7883a998aa70d611dc0698f87c1a3c52a50fb61e02 ./tests/lsps/LSP4DigitalAssetMetadata/LSP4Compatibility.test.ts

f212611c79c89a0e6a92310be41506bd005e1463685e9a051bfb90bff30fecad ./tests/lsps/LSP2ERC725YJSONSchema/LSP2UtilsLibrary.test.ts

75b29b64aa7ec2d9f7325d76c1b8eced0b909615b12288e12b996bb6c6ef445d ./tests/lsps/LSP1UniversalReceiver/LSP1UniversalReceiver.behaviour.ts

d7fe39dbfbe6ef39e18bbd881abde57e5b4a3d3b67f9fbd859434cb4e1fe4c5c
./tests/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP.behaviour.ts

83acb5b1ee677d694281cad215ab4ed97e826901b72f39e9e6e946245458a62c ./tests/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateUP.test.ts

96fe30e39185c333aa822836506babe93dcbbba120d24e823eb1eec7184996fc
./tests/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault.behaviour.ts

f01cb62c5fd4be89f369eb6f6f199b2d7fece154c5eb64f5689da85d4ebe450b ./tests/lsps/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault.test.ts

b41d5b18b61bc2de0241ff3a2b206de71fad6a747d211c0ef0fcf6b44a1eb1ec ./tests/lsps/Helpers/AddressRegistry.test.ts

ed01166a889d02eca2c8b9b3427dd2b81a99c0449ffc57a0caa5e7baf17082b5 ./tests/lsps/Helpers/ERC165CheckerCustom.test.ts

39f3c56cc08dc8892d423d17c6c3c07adf1cedda6e77dc9163cf47f81a772079 ./tests/lsps/Helpers/ERC165CheckerCustomTest.sol

073e3707404aafcc65011f7e4d05f607061411dad9e5adf1166e9f20a747471d ./tests/lsps/Helpers/ERC165Interfaces.sol

37b16f7eeb35b2e0ab201229b95f759ea6681bb75360aac1ce2315c3ddbba5c3 ./tests/lsps/Helpers/ERC165Interfaces.test.ts

70235360bc2e25d7ba63a9efba8542bd6a9827bb9fde115bb27308da496a3a76 ./tests/lsps/Helpers/Executor.sol

b72a48ed86ea00680d7c1378d909f2efcc8023aeb5ec86eadbd4e2c5a7659c52 ./tests/lsps/Helpers/FallbackContract.sol

1b24bfd844adeca499aa5024d640cb65ef9ffb9af298e102995438e027bcc052 ./tests/lsps/Helpers/ImplementationTester.sol

54b6ea417108dcd6518dc0b9965ca0c24b81158066df67ed29570cd8bf3002f4 ./tests/lsps/Helpers/KeyManagerExecutionCosts.test.ts

61799f50f8ab35213c00a461d2ddb44bd94c103b85ad9004c68ea06fe036602d ./tests/lsps/Helpers/LSP2UtilsLibraryTester.sol

b6f89ba973dbc11361a20c634bb4a7ea0b4edb211952c824ce8f897b857f5996 ./tests/lsps/Helpers/NFTStorage.sol

e2541495334c72325d1ecbe34aa2a3a88388475477219d292093acc13a888aba ./tests/lsps/Helpers/NFTStorage.test.ts

f0edc004c5c9244f53ab62991926100998e8a05335f51ef2637494e9dbda1f78 ./tests/lsps/Helpers/PayableContract.sol

cf8a77e9b37a2eea409c8746b629eb4cff727f328bb817e44657e2f632c58e56 ./tests/lsps/Helpers/SignatureValidatorContract.sol

5b2f6d9faf02b36e0a02e54762ad3fc6068835b17ec015c92eeae11dc3ce0e0e ./tests/lsps/Helpers/TargetContract.sol

aea7bff36d1744606ffc773d5aa6b5887241aeec310141f0f5805e3d37ca1588
./tests/lsps/Helpers/UniversalReceivers/UniversalReceiverDelegateTokenReentrant.sol

966278931cc283b335cfbe86f7d1f93cd11ebf265c5e190379c2dd6f82249f34
./tests/lsps/Helpers/UniversalReceivers/UniversalReceiverDelegateVaultSetter.sol

f5962830d2f0fb4a967537e6996f613d9f1e9f8ac53a40548193b6db1a6433f7 ./tests/lsps/Helpers/UniversalReceivers/UniversalReceiverTester.sol

1ed20dff4325c7079f7c8846c7b9074216dc8d91ef2a5ad8cf3751542cbf072f ./tests/lsps/Helpers/Tokens/IERC223.sol

94229c3743d208e01eedf787399874d574b4f6c36a6f3c0125452617b0adf272 ./tests/lsps/Helpers/Tokens/LSP4CompatibilityTester.sol

fc948a38fb5eac0d35de82f15a31f186a2d95eb868b38f708ad20b695e6b6a74 ./tests/lsps/Helpers/Tokens/LSP7CappedSupplyInitTester.sol

01b385b6e80ec869febe1d7ea89ca7299e19540fde35974ca83b7d11e044f2ac ./tests/lsps/Helpers/Tokens/LSP7CappedSupplyTester.sol

b09e31f2cd6674116cfa9a3fe72876eb204e923f9a6914a0142b545e57e3b870 ./tests/lsps/Helpers/Tokens/LSP7CompatibleERC20InitTester.sol

df493c4ca9a03771de4491ad09e410d47da87eadd4d8472dd37ac5da72eaf61a ./tests/lsps/Helpers/Tokens/LSP7CompatibleERC20Tester.sol

c528e535e880ccfde0316c5a133610091e1a45997f86b2ba8effebed89ebf296 ./tests/lsps/Helpers/Tokens/LSP7InitTester.sol

b6b84135ef17a5a90e792d93c3fc4d29e5bfe4dbf708db58df9b7177af4c5325 ./tests/lsps/Helpers/Tokens/LSP7Tester.sol

960472b2eb22ac2252884567527177cb698857a1baf497686a9f0a6310aac133 ./tests/lsps/Helpers/Tokens/LSP8CappedSupplyInitTester.sol

d838eca130c62cecb97da28d84fdc05ae549a3fa6f0c81d83ffb0cc772a163ec ./tests/lsps/Helpers/Tokens/LSP8CappedSupplyTester.sol

76f28959866b2d763b57e12fe6f40e2b67983ebc2adc6e98be725968964fdbc1 ./tests/lsps/Helpers/Tokens/LSP8CompatibleERC721Tester.sol

9e55ada2a136b9070229cf6f37bacea864f8f1ef84011c929edcdf701ed513a4 ./tests/lsps/Helpers/Tokens/LSP8CompatibleERC721TesterInit.sol

cfe24553e931e42be69bedf610151b215d659f6966338b5e04363565600532d9 ./tests/lsps/Helpers/Tokens/LSP8EnumerableInitTester.sol

b3f171f3b3e048f1be9de8c644132352c2b55c9d1eecf446cf3700922da9aa90 ./tests/lsps/Helpers/Tokens/LSP8EnumerableTester.sol

b02c46a56454e518a1ff6f0b89b8b76b0eb8484289b87547f95715afec9b76ac ./tests/lsps/Helpers/Tokens/LSP8InitTester.sol

70dfc7409299da4d9c3f1236393dcb04491f1984d5269eccb26ec258582d919f ./tests/lsps/Helpers/Tokens/LSP8Tester.sol

5fbbc3757128d5d4660e25a1a379c89d76ea0b5103dfd132eff6dae04fda506a ./tests/lsps/Helpers/Tokens/TokenReceiverWithLSP1.sol

738385d014e2f6672105fe9ed39cc4a127d34e5bcb02bef4798f74e37636789d ./tests/lsps/Helpers/Tokens/TokenReceiverWithoutLSP1.sol

b517c272bc23e2959939c21fce09cc123f1df8a6286630df9ef38d76963d6730 ./tests/lsps/Helpers/Security/Reentrancy.sol

0877c56750c9ddefec6eca213a25ed99a37ecf30fafddc9a772794b69ae57a38 ./tests/lsps/Helpers/KeyManager/ERC725YDelegateCall.sol

73489409f7a3d3931458ad42d8146207d8fb5c3b5fdda2d8da1be425ce840abe ./tests/lsps/Helpers/KeyManager/KeyManagerInternalsTester.sol

e8c2ce687db96e61b3ef02d4830fa853926d888cf529f0353bfba7be5a9d65ba ./tests/lsps/Helpers/KeyManager/TargetPayableContract.sol

81418a91e27217348dfb4316843fc18d473c654b3f0800679ee963c027290ab2 ./tests/lsps/Factories/UniversalFactory.test.ts

de3942566099210f68f7c060a5950013f2dfb74908ec7a06b325a815f6798929 ./tests/erc725/ERC165InterfaceId.test.ts

b733d52b7cbcfdf871ca4559b8602bb90a8a8aaec26e04579d0058b588ee1567 ./tests/erc725/ERC725.test.ts

6a55e7774a304919562e05b8b2cb982646a776b6fb21fb6b9a24ba9213c169b8 ./tests/erc725/ERC725X.behaviour.ts

cc98891bd88adfba6d82e6e410b61d239c016842cc10c4e67b573fe20e802c1f ./tests/erc725/ERC725X.test.ts

a1d1107df62defc7b147493c7fa32cc6299013d6906955b84341d0451e625207 ./tests/erc725/ERC725Y.behaviour.ts

0e3df5a60068a4e75a73d69283ae8dbb580066fc1d43acb080cfc9a7aec322a1 ./tests/erc725/ERC725Y.test.ts

249132c1d431b7355d4232220e852f2118e0fc784eeddb135aeb57b9f74c3d46 ./tests/erc725/fixtures.ts

88eed2f8ce41ac097d5a3b107d91b7604d3470d6a6e80df8cba760468e7646eb ./tests/erc725/helpers/Contract.sol

e71bb1e3eddefc6a42ce304dbfbaca4afdd13c73df52049150c5dfe5ad2ac1ad ./tests/erc725/helpers/Counter.sol

c6977cee193ecefc9202041f9cdcb33190910b569f7548ec3a0f8b595358a98b ./tests/erc725/helpers/CustomRevertTest.sol

51c4231a6269fda7b3a83af733f36685203632e07373a5666dfe7ed5fd2af8b2 ./tests/erc725/helpers/DelegateTest.sol

13d837339f4993da5ae06976263f4fdefb1754f96b8961036ab31e8adeaf7404 ./tests/erc725/helpers/ERC165InterfaceIDs.sol

aa9fd62ca8a2db999021c2410d33ae3f9ea18d8cc8e69cb3bf20b48499fef93b ./tests/erc725/helpers/ERC725XPayableTester.sol

4ca7af64441a7a4510bd05282234e6054a6360d4911272dce3124dd27db5bf05 ./tests/erc725/helpers/ERC725YReader.sol

e9b5ec73ef907919ac93c56efc34f68c291f145dd4cd3d89816b1708fe16ac86 ./tests/erc725/helpers/ERC725YWriter.sol

8f77534878b7659da6a2f912d8e539cec3a2c12588ff45e60c38c560c4ebc1c4 ./tests/erc725/helpers/NonPayableFallbackContract.sol

c61220abde4459f6a8673788dd24f178e718d75dc28366597df0743dc7689295 ./tests/erc725/helpers/NoReceive.sol

1ac1ab2bd686257ac1e890ac56b37072fd2768b08e229710cc33e40be8e2120e ./tests/erc725/helpers/PayableFallbackContract.sol

e95f9ec8343731813ab01275e517b9015be65b77d62e9e3822a0909af419a777 ./tests/erc725/helpers/Reader.sol

aa3a6e799c10c985743b4b1640722ecc6fc70612363f2e68c016c53653dec056 ./tests/erc725/helpers/ReceiveTester.sol

f6bc0b3eae59a953a6a10905d5dd393750ce5b7b7711fb33515b5355795b122f ./tests/erc725/helpers/Return.sol

2277b6b4599b3326cbba27c1a2fe50bd72981e7601e478f9b9824a3c7b04fad5 ./tests/erc725/helpers/selfdestruct.sol

460f7bcb7f1b744d6b9fb1860516aba16136bed316d0f0da243d1192a0ac89e5 ./tests/erc725/helpers/WithConstructorPayable.sol

ad7221a598c7015c17de24bcb6368b201810bcb241c10c35ceee4e895fdbb768 ./tests/erc725/helpers/WithConstructorWithArgs.sol

83d90e317bc037e92a4c63a9a03fa5cb8e6adc951cccf25b996d40ffca2969cf ./tests/erc725/helpers/WithConstructorWithoutArgs.sol

d1e4a8da541fece1f742dbfbb4eb9f0f7d1a2faad69346d22844baafa9651aa4 ./tests/erc725/helpers/WithoutConstructor.sol

Changelog

- 2022-08-10 - Initial report
- 2022-09-08 - Final report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.