



LUKSO LSPs #2 Audit Report

Dec 15, 2022



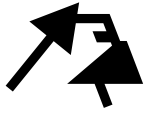
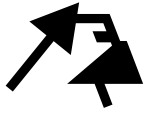


Table of Contents

Summary	2
Overview	3
Issues	4
[WP-M1] LSP-2: <code>LSP2Utils.isCompactByteArray(bytes)</code> Empty array (<code>[]</code>) will return <code>false</code>	4
[WP-M2] LSP-2: <code>LSP2Utils.isCompactByteArray(bytes)</code> does not support zero-length elements (<code>[..., 0x, ...]</code>)	6
[WP-D3] LSP-16: Consider adding a note about the requirements of <code>initializeCallData</code>	8
[WP-I4] LSP-16: Empty calldata (<code>fallback</code> , <code>receive</code>) for initialize is not differentiated with no initialize call	9
[WP-I5] LSP-8: <code>LSP8CompatibleERC721</code> is not compatible with <code>ERC721</code>	11
[WP-L6] LSP-14: <code>_pendingOwner</code> should be deleted when <code>_renounceOwnership</code> is called for the first time to initialize it	12
[WP-D7] LSP-2: Consider adding a note regarding the max element length in <code>bytes[CompactByteArray]</code>	16
[WP-I8] LSP-1: <code>LSP1UniversalReceiverDelegateUP</code> can be used for spamming	17
[WP-D9] LSP-2: Missing detailed documentation regarding the encoding of <code>bytesN[CompactByteArray]</code>	18
[WP-D10] LSP-9: The obscure <code>UPER_SETDATA</code> permission granted to <code>universalReceiverDelegate</code> should be explicitly documented	19
[WP-L11] LSP-6: <code>SuperTransferValue</code> permission can be used to initiate calls	23
[WP-L12] LSP-6: <code>_verifyCanExecute</code> may unexpectedly allow function calls	24
[WP-L13] LSP-6: <code>AddressPermissions.AllowedCalls:<address></code> should use <code>bytes28[]</code> or <code>bytes28[CompactByteArray]</code> rather than <code>bytes[CompactByteArray]</code>	25
[WP-I14] LSP-17: The abstract contract <code>LSP17Extension</code> should include access control to avoid misapplication	27

[WP-I15] LSP-17: Unconventional behavior: when called with an unsupported <code>msg.sig</code> , <code>fallback()</code> does not revert	30
[WP-D16] LSP-17: Consider explicitly document that <code>_fallbackLSP17Extendable()</code> should be called at the end of the <code>fallback()</code> function	33
[WP-I17] LSP-14: <code>renounceOwnership()</code> is considered unnecessary and error-prone for LSP-0 and LSP-9	35
[WP-D18] LSP-14: Event name in the implementation does not match the documentation	36
[WP-I19] LSP-14: Using number of block for CONFIRMATION_DELAY will result in different length of time in different networks	38
[WP-G20] LSP-14: Unnecessary SLOAD	40
[WP-G21] LSP-1: Check the <code>notifierMapValue</code> first can save gas	42
[WP-N22] LSP-6: Inconsistent <code>InvalidCompactByteArrayLengthElement</code> check	45
[WP-D23] LSP-6: The corresponding <code>valueContent</code> of <code>valueType</code> <code>bytes[CompactByteArray]</code> should be <code>bytes[]</code> instead of <code>Bytes</code>	47
Appendix	49
Disclaimer	50



Summary

This report has been prepared for LUKSO LSPs #2 Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.



Overview

Project Summary

Project Name	LUKSO LSPs #2 Audit Report
Codebase	https://github.com/lukso-network/lsp-smart-contracts
Commit	e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2
Language	Solidity

Audit Summary

Delivery Date	Dec 15, 2022
Audit Methodology	Static Analysis, Manual Review
Total Issues	23



Medium

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol#L266-L297>

5

```
289      */
290      while (pointer < compactByteArray.length) {
291          uint256 elementLength =
292          uint256(uint8(bytes1(compactByteArray[pointer])));
293          if (elementLength == 0) return false;
294          pointer += elementLength + 1;
295      }
296      if (pointer == compactByteArray.length) return true;
297      return false;
298  }
```

Impact

This can be a problem when trying to reset the value of a `bytes[CompactByteArray]` to empty array.

Such a problem does not exist in the current LSP-6 implementation, where `bytes[CompactByteArray]` is used.

However, given the fact that LSP-2 is designed to be a general-purpose library, we still consider this a medium severity issue and should be addressed.

Recommendation

Consider removing L270.

[WP-M2] LSP-2: `LSP2Utils.isCompactByteArray(bytes)` does not support zero-length elements (`[..., 0x, ...]`)

Medium

Issue Description

`bytes[CompactByteArray]` should support zero-length elements (`0x`).

However, the current implementation of `LSP2Utils.isCompactByteArray(bytes)` will return `false` whenever there is a zero-length element.

PoC

When using `isCompactByteArray()` to verify `0x 00 03 222222` (`[0x, 0x222222]` encoded in `bytes[CompactByteArray]`)

L292 will return `false` .

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol#L266-L297>

```

266      /**
267      * @dev Verify the validity of the `compactByteArray` according to LSP2
268      */
269      function isCompactByteArray(bytes memory compactByteArray) internal pure
returns (bool) {
270          if (compactByteArray.length == 0) return false;
271      /**
272      * Pointer will always land on these values:
273      *
274      * ↓↓
275      * 03 a00000
276      * 05 fff83a0011
277      * 20 aa0000000000000000000000000000000000000000000000000000000000000000cafe
278      * 12 bb00000000000000000000000000000000000000000000000000000000000000beef
279      * 19 cc00000000000000000000000000000000000000000000000000000000000000deed
280      * ↑↑
281      */

```



```

282         * The pointer can only land on the length of the following bytes value.
283         */
284         uint256 pointer;
285
286         /**
287         * Check each Length byte and make sure that when you reach the Last
288         * Length byte.
289         * Make sure that the Last Length describes exactly the last bytes value
290         * and you do not get out of bounds.
291         */
292         while (pointer < compactByteArray.length) {
293             uint256 elementLength =
294             uint256(uint8(bytes1(compactByteArray[pointer])));
295             if (elementLength == 0) return false;
296             pointer += elementLength + 1;
297         }
298         if (pointer == compactByteArray.length) return true;
299         return false;
300     }

```

Impact

Certain use cases will be disabled, such as:

- Using the length of a `CompactByteArray` to represent the number of available seats, with empty elements for available seats and elements with a value representing an order ID.
- Using empty elements as placeholders to maintain the index of pre-existing elements.

Recommendation

Consider removing L292.

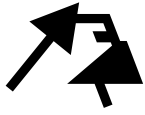


[WP-D3] LSP-16: Consider adding a note about the requirements of `initializeCallData`

Issue Description

When `initializeCallData` includes non-crosschain parameters, the deployed contract will not be recreated at the same address on another network with the same calldata, thus defeating the purpose of `LSP16UniversalFactory`.

Therefore, `initializeCallData` must not include any network-specific parameters, such as a local non-crosschain token contract address.



[WP-I4] LSP-16: Empty calldata (`fallback` , `receive`) for `initialize` is not differentiated with no `initialize` call

Informational

Issue Description

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP16UniversalFactory/LSP16UniversalFactory.sol#L191-L202>

```
191 function _generateSalt(bytes memory initializeCallData, bytes32 providedSalt)
192     internal
193     pure
194     returns (bytes32)
195 {
196     bool initializable = initializeCallData.length != 0;
197     if (initializable) {
198         return keccak256(abi.encodePacked(initializable, initializeCallData,
199             providedSalt));
200     } else {
201         return keccak256(abi.encodePacked(initializable, providedSalt));
202     }
203 }
```


When the length of `initializeCallData` is `0` , it will not be included in the salt.

As a result, a special Clone with no `initializeCallData` but an empty call (`contractCreated.call()`) is indistinguishable to a `Create2` without that empty initialize call.

Although it is highly unlikely that a proxy contract would be designed to be initialized with an empty call, it is technically possible and permissible.

Recommendation

We believe it is more consistent and unbiased to allow empty initialize calls, therefore, we recommend you to make the following changes:

- 
1. `_generateSalt()` should differentiate between no initialize call and an initialize call with no `initializeCallData` ;
 2. `deployCreate2Proxy()` should be split into two functions: `deployCreate2Proxy()` and `deployCreate2ProxyInit()` ;
 3. Line 128, `if (initializeCalldata.length == 0) revert InitializeCalldataRequired();` , should be removed to allow empty initialize calls.

[WP-I5] LSP-8: `LSP8CompatibleERC721` is not compatible with `ERC721`

Informational

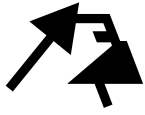
Issue Description

This is an extension of [WP-I17] from our previous report.

- `safeTransferFrom()` does not call `receiver.onERC721Received(address,address,uint256,bytes)` which is required by EIP-721's spec, and does not require `receiver.onERC721Received(...) == bytes4(keccak256("onERC721Received(address,address,uint256,bytes4)"))` ;
- `transferFrom()` also calls `_notifyTokenSender()` , `_notifyTokenReceiver()` ;
- `safeTransferFrom()` also calls `_notifyTokenSender()` .

Recommendation

1. Consider improving the compatibility of `LSP8CompatibleERC721` to ERC721.
2. Or, consider documenting the differences between `LSP8CompatibleERC721` and ERC721.



[WP-L6] LSP-14: `_pendingOwner` should be deleted when `_renounceOwnership` is called for the first time to initialize it

Low

Issue Description

Otherwise, two bad scenarios might occur:

PoC #1:

1. Current owner Bob first calls `transferOwnership()` to set Alice as the `pendingOwner`.
2. Bob changes his mind and calls `_renounceOwnership()` to initiate giving up ownership.
3. Before Bob calls `_renounceOwnership()` for the second time to finalize the renounce, Alice calls `acceptOwnership()`, which is not in line with Bob's expectation.

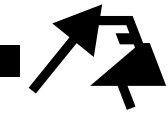
PoC #2:

Building on PoC 1, after Alice becomes the owner and wants to renounce the ownership, she calls `_renounceOwnership()`.

If the call happens to fall between `confirmationPeriodStart` and `confirmationPeriodEnd` triggered by Bob, the expectation is to initiate the renouncement. However, the actual result is that the renouncement is done immediately and the owner will become `address(0)` right after the call.

<https://github.com/lukso-network/lsp-smart-contracts/blob/848026307f597f42a8a769b3a4b4152a0907e2ed/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162>

```
141     function _renounceOwnership() internal virtual {
142         uint256 currentBlock = block.number;
143         uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
144             _RENOUCE_OWNERSHIP_CONFIRMATION_DELAY;
145         uint256 confirmationPeriodEnd = confirmationPeriodStart +
146             _RENOUCE_OWNERSHIP_CONFIRMATION_PERIOD;
147
148         if (currentBlock > confirmationPeriodEnd) {
149             _renounceOwnershipStartedAt = currentBlock;
```



```

150         emit RenounceOwnershipInitiated();
151         return;
152     }
153
154     if (currentBlock < confirmationPeriodStart) {
155         revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
156     }
157
158     _setOwner(address(0));
159     delete _renounceOwnershipStartedAt;
160     delete _pendingOwner;
161     emit OwnershipRenounced();
162 }

```

Recommendation

We've come up with 2 possible resolutions for this:

1. Consider introducing a lock to prevent transfer of ownership during the renouncement of ownership and vice-versa:

```

141 function _renounceOwnership() internal virtual {
142     require(_pendingOwner == address(0), "_renounceOwnership is not allowed
now.");
143     uint256 currentBlock = block.number;
144     uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
_RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
145     uint256 confirmationPeriodEnd = confirmationPeriodStart +
_RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;
146
147     if (currentBlock > confirmationPeriodEnd) {
148
149         _renounceOwnershipStartedAt = currentBlock;
150         emit RenounceOwnershipInitiated();
151         return;
152     }
153
154     if (currentBlock < confirmationPeriodStart) {
155         revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
156     }
157 }

```

```

158
159     _setOwner(address(0));
160     delete _renounceOwnershipStartedAt;
161     emit OwnershipRenounced();
162 }

```

```

100 function _transferOwnership(address newOwner) internal virtual {
101     require(_renounceOwnershipStartedAt == 0, "_transferOwnership is not allowed
now.");
102     if (newOwner == address(this)) revert CannotTransferOwnershipToSelf();
103
104     _pendingOwner = newOwner;
105     address currentOwner = owner();
106     emit OwnershipTransferStarted(currentOwner, newOwner);
107
108     _notifyUniversalReceiver(newOwner, _TYPEID_LSP14_OwnershipTransferStarted,
    "");
109     require(
110         currentOwner == owner(),
111         "LSP14: newOwner MUST accept ownership in a separate transaction"
112     );
113 }

```

1. The initialization of `transferOwnership` and `renounceOwnership` will cancel each other:

```

141 function _renounceOwnership() internal virtual {
142     uint256 currentBlock = block.number;
143     uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
144         _RENOUCE_OWNERSHIP_CONFIRMATION_DELAY;
145     uint256 confirmationPeriodEnd = confirmationPeriodStart +
146         _RENOUCE_OWNERSHIP_CONFIRMATION_PERIOD;
147
148     if (currentBlock > confirmationPeriodEnd) {
149         _renounceOwnershipStartedAt = currentBlock;
150         // cancel _transferOwnership if any
151         delete _pendingOwner;
152         emit RenounceOwnershipInitiated();
153         return;
154     }
155 }

```



```
156     if (currentBlock < confirmationPeriodStart) {
157         revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
158     }
159
160     _setOwner(address(0));
161     delete _renounceOwnershipStartedAt;
162     emit OwnershipRenounced();
163 }
```

```
100 function _transferOwnership(address newOwner) internal virtual {
101     if (newOwner == address(this)) revert CannotTransferOwnershipToSelf();
102
103     _pendingOwner = newOwner;
104     // cancel _renounceOwnership if any
105     delete _renounceOwnershipStartedAt;
106     address currentOwner = owner();
107     emit OwnershipTransferStarted(currentOwner, newOwner);
108
109     _notifyUniversalReceiver(newOwner, _TYPEID_LSP14_OwnershipTransferStarted,
110     "");
111     require(
112         currentOwner == owner(),
113         "LSP14: newOwner MUST accept ownership in a separate transaction"
114     );
115 }
```

[WP-D7] LSP-2: Consider adding a note regarding the max element length in `bytes[CompactByteArray]`

Issue Description

The maximum length of each element is 255, because `uint8` is used to store the length of each element and the maximum value of `uint8` is 255.

See:

<https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-2-ERC725YJSONSchema.md#bytescompactbytearray>

This should be explicitly documented to avoid misapplication.



[WP-I8] LSP-1: `LSP1UniversalReceiverDelegateUP` can be used for spamming

Informational

Issue Description

Due to the permissionless and open design of `LSP1UniversalReceiverDelegateUP`, there is a potential for malicious or spam assets to be transferred to the target UP, thus cluttering the list of received assets.

For instance:

- Malicious NFTs with Trojan viruses embedded in the images;
- Scam tokens.

Recommendation

1. Adding a filter of blocklist on the frontend level to block out malicious tokens/NFTs;
2. Providing a helper dapp to quickly remove a set of undesired tokens/NFTs.

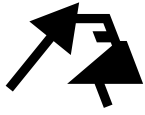


[WP-D9] LSP-2: Missing detailed documentation regarding the encoding of `bytesN[CompactByteArray]`

Issue Description

Based on the context, the encoding of `bytesN[CompactByteArray]` must be different from `bytes[CompactByteArray]` and `bytesN[]`.

However, the detailed documentation on the encoding of `bytesN[CompactByteArray]` is missing.



[WP-D10] LSP-9: The obscure `UPER_SETDATA` permission granted to `universalReceiverDelegate` should be explicitly documented

Issue Description

The `universalReceiverDelegate` of LSP9Vault will be granted a temporary `_reentrantDelegate` role with unlimited `setData()` permission during the incoming `universalReceiver()` call.

If an LSP-6 is the owner of the LSP9Vault, then such a permission will be equivalent to the `SUPER_SETDATA` permission.

Plus, `universalReceiver()` is a public function that can be called by anyone at any time.

This allows a malicious or compromised `universalReceiverDelegate` to do many things that are considered dangerous, eg:

1. Adding or updating a `extension` via LSP-17;
2. Changing the permission via LSP-6 (if the KM is the owner).

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L229-L280>

```
229 function universalReceiver(bytes32 typeId, bytes calldata receivedData)
230     public
231     payable
232     virtual
233     returns (bytes memory returnedValues)
234 {
235     if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
236     bytes memory lsp1DelegateValue =
        _getData(_LSP1_UNIVERSAL_RECEIVER_DELEGATE_KEY);
237     bytes memory resultDefaultDelegate;
238
239     if (lsp1DelegateValue.length >= 20) {
240         address universalReceiverDelegate = address(bytes20(lsp1DelegateValue));
241
242         if (universalReceiverDelegate.supportsERC165Interface(_INTERFACEID_LSP1))
243         {
244             _reentrantDelegate = universalReceiverDelegate;
245         }
246     }
247 }
```

```

244         resultDefaultDelegate = universalReceiverDelegate
245             .callUniversalReceiverWithCallerInfos(
246                 typeId,
247                 receivedData,
248                 msg.sender,
249                 msg.value
250             );
251     }
252 }
253
@@ 254,260 @@
261
262     if (lsp1TypeIdDelegateValue.length >= 20) {
263         address universalReceiverDelegate =
264         address(bytes20(lsp1TypeIdDelegateValue));
265
266         if (universalReceiverDelegate.supportsERC165Interface(_INTERFACEID_LSP1))
267         {
268             _reentrantDelegate = universalReceiverDelegate;
269             resultTypeIdDelegate = universalReceiverDelegate
270                 .callUniversalReceiverWithCallerInfos(
271                     typeId,
272                     receivedData,
273                     msg.sender,
274                     msg.value
275                 );
276         }
277     }
278
279     delete _reentrantDelegate;
280     returnedValues = abi.encode(resultDefaultDelegate, resultTypeIdDelegate);
281     emit UniversalReceiver(msg.sender, msg.value, typeId, receivedData,
282         returnedValues);
283 }

```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L192-L194>

```

192 function setData(bytes32 dataKey, bytes memory dataValue) public virtual override
    onlyAllowed {
193     _setData(dataKey, dataValue);
194 }

```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L204-L217>

```

204 function setData(bytes32[] memory dataKeys, bytes[] memory dataValues)
205     public
206     virtual
207     override
208     onlyAllowed
209 {
210     if (dataKeys.length != dataValues.length) {
211         revert ERC725Y_DataKeysValuesLengthMismatch(dataKeys.length,
dataValues.length);
212     }
213
214     for (uint256 i = 0; i < dataKeys.length; i = GasLib.uncheckedIncrement(i)) {
215         _setData(dataKeys[i], dataValues[i]);
216     }
217 }

```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L64-L75>

```

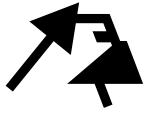
64 /**
65  * @dev Modifier restricting the call to the owner of the contract and the
    UniversalReceiverDelegate
66  */
67 modifier onlyAllowed() {
68     if (msg.sender != owner()) {
69         require(
70             msg.sender == _reentrantDelegate,
71             "Only Owner or reentered Universal Receiver Delegate allowed"
72         );

```

```
73         }  
74     _;  
75 }
```

Recommendation

Considering the high impact of this implied `SUPER_SETDATA` permission to `universalReceiverDelegate` , it should either be restricted in some way or explicitly documented in the documentation.



[WP-L11] LSP-6: `SuperTransferValue` permission can be used to initiate calls

Low

Issue Description

`SuperTransferValue` permission should differentiate between:

A, transfers of native tokens to EOA, or contracts with no code execution in the `recieve()` / `fallback()` function; B, contracts that do have code execution in the `recieve()` / `fallback()` function.

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L828-L830>

```
828 // Skip if caller has SUPER permission for value transfers
829 if (hasSuperTransferValue && !isCallDataPresent && value != 0) return;
```

Recommendation

We believe that when the target is a smart contract (`.code.length > 0`), it should require the `CALL` permission, or even the `SUPERCALL` permission.



[WP-L12] LSP-6: `_verifyCanExecute` may unexpectedly allow function calls

Low

Issue Description

`isCallDataPresent` and `containsFunctionCall` are determined based on the length of the payload.

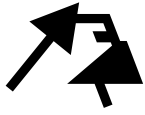
One can bypass this by using `receive()` / `fallback()` to execute code on the target contract.

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L842-L845>

```
842  bool containsFunctionCall = payload.length >= 168;  
843  bytes4 selector;  
844  if (containsFunctionCall) selector = bytes4(payload[164:168]);
```

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L805>

```
805  bool isCallDataPresent = payload.length > 164;
```



[WP-L13] LSP-6: `AddressPermissions:AllowedCalls:<address>` should use `bytes28[]` or `bytes28[CompactByteArray]` rather than `bytes[CompactByteArray]`

Low

Issue Description

Per the docs and the implementation:

The compact bytes array MUST be constructed in this format according to [LSP2-ERC725YJSONSchema]:

`<1c> <bytes4 allowedInterfaceId> <bytes20 allowedAddress> <bytes4 allowedFunction> 1c:`
1c in decimals is 28, which is the sum of bytes length of the elements stored in the array.

Thus, we believe `bytes28[]` or `bytes28[CompactByteArray]` would be more suitable for this.

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L838-L877>

```
838 function _verifyAllowedCall(address from, bytes calldata payload) internal view {
839     // CHECK for ALLOWED CALLS
840     address to = address(bytes20(payload[48:68]));
841
842     bool containsFunctionCall = payload.length >= 168;
843     bytes4 selector;
844     if (containsFunctionCall) selector = bytes4(payload[164:168]);
845
846     bytes memory allowedCalls = ERC725Y(target).getAllowedCallsFor(from);
847     uint256 allowedCallsLength = allowedCalls.length;
848
849     if (allowedCallsLength == 0 || !LSP2Utils.isCompactByteArray(allowedCalls)) {
850         revert NoCallsAllowed(from);
851     }
852
853     bool isAllowedStandard;
```

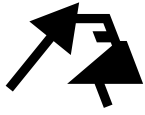
```

854     bool isAllowedAddress;
855     bool isAllowedFunction;
856
857     for (uint256 ii = 0; ii < allowedCallsLength; ii += 29) {
858
859         bytes memory chunk = BytesLib.slice(allowedCalls, ii + 1, 28);
860
861         if (bytes28(chunk) ==
862             0xffffffffffffffffffffffffffffffffffffffffffffffffffffffff) {
863             revert InvalidWhitelistedCall(from);
864         }
865
866         bytes4 allowedStandard = bytes4(chunk);
867         address allowedAddress = address(bytes20(bytes28(chunk) << 32));
868         bytes4 allowedFunction = bytes4(bytes28(chunk) << 192);
869
870         isAllowedStandard = allowedStandard == 0xffffffff ||
871         to.supportsERC165Interface(allowedStandard);
872         isAllowedAddress = allowedAddress ==
873         0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF || to == allowedAddress;
874         isAllowedFunction = allowedFunction == 0xffffffff || containsFunctionCall
875         && (selector == allowedFunction);
876
877         if (isAllowedStandard && isAllowedAddress && isAllowedFunction) return;
878     }
879
880     revert NotAllowedCall(from, to, selector);
881 }

```

Recommendation

Consider using `bytes28[]` or `bytes28[CompactByteArray]` instead.



[WP-I14] LSP-17: The abstract contract `LSP17Extension` should include access control to avoid misapplication

Informational

Issue Description

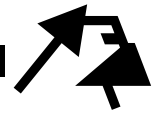
The concrete contracts that inherit `LSP17Extension` will most certainly have some privileges on the LSP-0.

Thus, it must include some sort of access control to avoid unpermitted calls.

<https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-17-ContractExtension.md#security-considerations>

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP17ContractExtension/LSP17Extension.sol#L14-L45>

```
14  abstract contract LSP17Extension is ERC165 {
15      // solhint-disable
16
17      /**
18       * @dev See {IERC165-supportsInterface}.
19       */
20      function supportsInterface(bytes4 interfaceId) public view virtual override
21      returns (bool) {
22          return interfaceId == _INTERFACEID_LSP17_EXTENSION ||
23          super.supportsInterface(interfaceId);
24      }
25
26      /**
27       * @dev Returns the original msg.data passed to the extendable contract
28       * without the appended msg.sender and msg.value
29       */
30      function _extendableMsgData() internal view virtual returns (bytes calldata) {
31          return msg.data[:msg.data.length - 52];
32      }
33
34      /**
```



```

33     * @dev Returns the original msg.sender calling the extendable contract
34     */
35     function _extendableMsgSender() internal view virtual returns (address) {
36         return address(bytes20(msg.data[msg.data.length - 52:msg.data.length -
37     32]));
38     }
39     /**
40     * @dev Returns the original msg.value sent to the extendable contract
41     */
42     function _extendableMsgValue() internal view virtual returns (uint256) {
43         return uint256(bytes32(msg.data[msg.data.length - 32:]));
44     }
45 }

```

Recommendation

The abstract contract should provide a framework for access control:

```

10  /**
11   * @title Implementation of the extension Logic according to
12   * @dev To be inherited to provide context of the msg variable related to the
13   * @dev extendable contract
14   */
15  abstract contract LSP17Extension is ERC165 {
16      /// @custom:oz-upgrades-unsafe-allow state-variable-immutable
17      address private immutable _trustedCaller;
18
19      /// @custom:oz-upgrades-unsafe-allow constructor
20      constructor(address trustedCaller) {
21          _trustedCaller = trustedCaller;
22      }
23
24      function isTrustedCaller(address caller) public view virtual returns (bool) {
25          return caller == _trustedCaller;
26      }
27
28      /**
29       * @dev See {IERC165-supportsInterface}.

```

```

30     */
31     function supportsInterface(bytes4 interfaceId) public view virtual override
returns (bool) {
32         return interfaceId == _INTERFACEID_LSP17_EXTENSION ||
super.supportsInterface(interfaceId);
33     }
34
35     /**
36     * @dev Returns the original msg.data passed to the extendable contract
37     * without the appended msg.sender and msg.value
38     */
39     function _extendableMsgData() internal view virtual returns (bytes calldata) {
40         if (isTrustedCaller(msg.sender)) {
41             return msg.data[:msg.data.length - 52];
42         } else {
43             return msg.data;
44         }
45     }
46
47     /**
48     * @dev Returns the original msg.sender calling the extendable contract
49     */
50     function _extendableMsgSender() internal view virtual returns (address) {
51         if (isTrustedCaller(msg.sender)) {
52             return address(bytes20(msg.data[msg.data.length - 52:msg.data.length -
32]));
53         } else {
54             return msg.sender;
55         }
56     }
57
58     /**
59     * @dev Returns the original msg.value sent to the extendable contract
60     */
61     function _extendableMsgValue() internal view virtual returns (uint256) {
62         if (isTrustedCaller(msg.sender)) {
63             return uint256(bytes32(msg.data[msg.data.length - 32:]));
64         } else {
65             return msg.value;
66         }
67     }
68 }

```

[WP-I15] LSP-17: Unconventional behavior: when called with an unsupported `msg.sig`, `fallback()` does not revert

Informational

Issue Description

A regular smart contract will revert when called with an unsupported `msg.sig`.

Therefore, for the caller, a common way to confirm a successful call is: `.code.length > 0` && the contract call did not revert (such as OpenZeppelin's implementation).

Lukso breaks this assumption (no revert when encountering an unsupported `msg.sig`), which leads to misjudgment of the above method.

For example, OpenZeppelin's `does not revert` as expected.

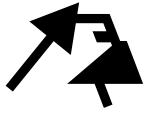
Similar to LSP-17, the Diamond proxy states explicitly in the spec that execution should revert when an unsupported function call is encountered.

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L121-L124>

```
121     fallback() external payable virtual {
122         if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
123         _fallbackLSP17Extendable();
124     }
```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP0ERC725Account/LSP0ERC725AccountCore.sol#L109-L112>

```
109     fallback() external payable virtual {
110         if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
111         _fallbackLSP17Extendable();
112     }
```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP17ContractExtension/LSP17Extendable.sol#L58-L94>

```
58     function _fallbackLSP17Extendable() internal virtual {
59         if (msg.data.length < 4) return;
60         // If there is a function selector
61         address extension = _getExtension(msg.sig);
62
63         // if no extension was found, return
64         if (extension == address(0)) return;
65
66         @@ 66,93 @@
94     }
```

Recommendation

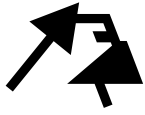
Consider reverting when in the case of an unsupported `msg.sig` .

Here is an illustrative example of how a diamond's fallback function might be implemented:

```
1  // Find facet for function that is called and execute the
2  // function if a facet is found and return any value.
3  fallback() external payable {
4      // get facet from function selector
5      address facet = selectorTofacet[msg.sig];
6      require(facet != address(0));
7      // Execute external function from facet using delegatecall and return any value.
8      assembly {
9          // copy function selector and any arguments
10         calldatacopy(0, 0, calldatasize())
11         // execute function call using the facet
12         let result := delegatecall(gas(), facet, 0, calldatasize(), 0, 0)
13         // get any return value
14         returndatacopy(0, 0, returndatasize())
15         // return any return value or error back to the caller
16         switch result
17         case 0 {revert(0, returndatasize())}
18         default {return (0, returndatasize())}
19     }
```



20 }




[WP-D16] LSP-17: Consider explicitly document that `_fallbackLSP17Extendable()` should be called at the end of the `fallback()` function

Issue Description

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP17ContractExtension/LSP17Extendable.sol#L58-L94>

```
58  function _fallbackLSP17Extendable() internal virtual {
59      if (msg.data.length < 4) return;
60      // If there is a function selector
61      address extension = _getExtension(msg.sig);
62
63      // if no extension was found, return
64      if (extension == address(0)) return;
65
66      // solhint-disable no-inline-assembly
67      // if the extension was found, call the extension with the msg.data
68      // appended with bytes20(address) and bytes32(msg.value)
69      assembly {
70          calldatacopy(0, 0, calldatasize())
71
72          // The msg.sender address is shifted to the left by 12 bytes to remove the
padding
73          // Then the address without padding is stored right after the calldata
74          mstore(calldatasize(), shl(96, caller()))
75
76          // The msg.value is stored right after the calldata + msg.sender
77          mstore(add(calldatasize(), 20), callvalue())
78
79          // Add 52 bytes for the msg.sender and msg.value appended at the end of
the calldata
80          let success := call(gas(), extension, 0, 0, add(calldatasize(), 52), 0, 0)
81
82          // Copy the returned data
83          returndatacopy(0, 0, returndatasize())
84
85          switch success
```



```
86      // call returns 0 on failed calls
87      case 0 {
89          revert(0, returndatasize())
90      }
91      default {
92          return(0, returndatasize())
93      }
94  }
```

As `_fallbackLSP17Extendable()` uses assembly `return()` / `revert()` to terminate the call, it cannot be called before other codes in `fallback()` .

Otherwise, the codes after `_fallbackLSP17Extendable()` may never be reached.

[WP-I17] LSP-14: `renounceOwnership()` is considered unnecessary and error-prone for LSP-0 and LSP-9

Informational

Issue Description

`renounceOwnership()` is a function provided by `LSP14Ownable2Step` and `OwnableUnset`.

It can be useful for a general smart contract with a few `onlyOwner` functions that are only used during the initializing period and can be abandoned afterwards to avoid centralization risks.

However, both LSP-0 and LSP-9 rely on the owner for their core features.

We believe there is no use case where `renounceOwnership()` can be helpful.

Furthermore, once `renounceOwnership()` is called, the caller will immediately lose control over the LSP0/LSP9 and the error is irreversible.

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP9Vault/LSP9VaultCore.sol#L296-L306>

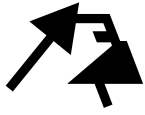
```

296     /**
297      * @dev Renounce ownership of the contract in a 2-step process
298      */
299     function renounceOwnership()
300         public
301         virtual
302         override(LSP14Ownable2Step, OwnableUnset)
303         onlyOwner
304     {
305         LSP14Ownable2Step._renounceOwnership();
306     }

```

Recommendation

Consider overriding and reverting in `renounceOwnership()`.



[WP-D18] LSP-14: Event name in the implementation does not match the documentation

Issue Description

In the docs

```
1  event RenounceOwnershipStarted();
```

MUST be emitted when the process of renouncing ownership of the contract is initiated.

The same event is called `RenounceOwnershipInitiated` in the implementation:

<https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L35-L38>

```
35  /**
36   * @dev emitted whenever the `renounceOwnership(..)` 2-step process is started
37   */
38  event RenounceOwnershipInitiated();
```

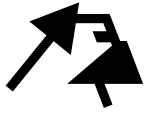
<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162>

```
141  function _renounceOwnership() internal virtual {
142      uint256 currentBlock = block.number;
143      uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
144          _RENOUCE_OWNERSHIP_CONFIRMATION_DELAY;
145      uint256 confirmationPeriodEnd = confirmationPeriodStart +
146          _RENOUCE_OWNERSHIP_CONFIRMATION_PERIOD;
147
148      if (currentBlock > confirmationPeriodEnd) {
149          _renounceOwnershipStartedAt = currentBlock;
150          emit RenounceOwnershipInitiated();
```

```
151         return;
152     }
153
154     if (currentBlock < confirmationPeriodStart) {
155         revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
156         confirmationPeriodEnd);
157     }
158     _setOwner(address(0));
159     delete _renounceOwnershipStartedAt;
160     delete _pendingOwner;
161     emit OwnershipRenounced();
162 }
```

Recommendation

Based on the context, we believe the name `RenounceOwnershipStarted()` in the docs is incorrect and should be changed to `RenounceOwnershipInitiated` .



[WP-I19] LSP-14: Using number of block for CONFIRMATION_DELAY will result in different length of time in different networks

Informational

Issue Description

Because different chains have different block time.

Also, it is not convenient to query CONFIRMATION_DELAY and CONFIRMATION_PERIOD with private visibility.

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L45-L54>

```
45      /**
46       * @dev The number of block that MUST pass before one is able to
47       * confirm renouncing ownership
48       */
49      uint256 private constant _RENOUCE_OWNERSHIP_CONFIRMATION_DELAY = 100;
50
51      /**
52       * @dev The number of blocks during which one can renounce ownership
53       */
54      uint256 private constant _RENOUCE_OWNERSHIP_CONFIRMATION_PERIOD = 100;
```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162>

```
141      function _renounceOwnership() internal virtual {
142          uint256 currentBlock = block.number;
143          uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
144              _RENOUCE_OWNERSHIP_CONFIRMATION_DELAY;
145          uint256 confirmationPeriodEnd = confirmationPeriodStart +
146              _RENOUCE_OWNERSHIP_CONFIRMATION_PERIOD;
```



```

147
148     if (currentBlock > confirmationPeriodEnd) {
149         _renounceOwnershipStartedAt = currentBlock;
150         emit RenounceOwnershipInitiated();
151         return;
152     }
153
154     if (currentBlock < confirmationPeriodStart) {
155         revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
156     }
157
158     _setOwner(address(0));
159     delete _renounceOwnershipStartedAt;
160     delete _pendingOwner;
161     emit OwnershipRenounced();
162 }

```

Recommendation

1. Consider changing to public visibility

2. `renounceOwnership`

`renounceOwnership` number of block `confirmationPeriod` step duration

or

`renounceOwnership` immutable `confirmationPeriod`

[WP-G20] LSP-14: Unnecessary SLOAD

Gas

Issue Description

The SLOAD of `_pendingOwner` at L122 is unnecessary, as it must be `msg.sender`.

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L118-L135>

```


118  function _acceptOwnership() internal virtual {
119      require(msg.sender == pendingOwner(), "LSP14: caller is not the
120             pendingOwner");
121
122      address previousOwner = owner();
123      _setOwner(_pendingOwner);
124      delete _pendingOwner;
125
126      _notifyUniversalReceiver(
127          previousOwner,
128          _TYPEID_LSP14_OwnershipTransferred_SenderNotification,
129      );
130      _notifyUniversalReceiver(
131          msg.sender,
132          _TYPEID_LSP14_OwnershipTransferred_RecipientNotification,
133      );
134  }
135  }
```

Recommendation

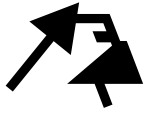
Change to:

```

118  function _acceptOwnership() internal virtual {
119      require(msg.sender == pendingOwner(), "LSP14: caller is not the
120             pendingOwner");
121  }
```



```
121     address previousOwner = owner();
122     _setOwner(msg.sender);
123     delete _pendingOwner;
124
125     _notifyUniversalReceiver(
126         previousOwner,
127         _TYPEID_LSP14_OwnershipTransferred_SenderNotification,
128         ""
129     );
130     _notifyUniversalReceiver(
131         msg.sender,
132         _TYPEID_LSP14_OwnershipTransferred_RecipientNotification,
133         ""
134     );
135 }
```



[WP-G21] LSP-1: Check the `notifierMapValue` first can save gas

Gas

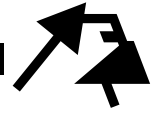
Issue Description

When `bytes12(notifierMapValue) != bytes12(0)`, the rather expensive external call of `ILSP7DigitalAsset(notifier).balanceOf(msg.sender)` is unnecessary.

Moving L61-62 down to after L66 can save gas from the unnecessary external call in such case.

<https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/LSP1UniversalReceiverDelegateVault.sol#L38-L85>

```
38  function universalReceiver(  
39      bytes32 typeId,  
40      bytes memory /* data */  
41  ) public payable virtual returns (bytes memory result) {  
42      if (msg.value != 0) revert NativeTokensNotAccepted();  
43      // This contract acts like a UniversalReceiverDelegate of a Vault where we  
44      // append the  
45      // address and the value, sent to the universalReceiver function of the LSP9,  
46      // to the msg.data  
47      // Check  
48      // https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-9-Vault.md#universalreceiver  
49      address notifier = address(bytes20(msg.data[msg.data.length - 52:]));  
50  
51      (bool invalid, bytes10 mapPrefix, bytes4 interfaceID, bool isReceiving) =  
52      LSP1Utils  
53      .getTransferDetails(typeId);  
54  
55      if (invalid || interfaceID == _INTERFACEID_LSP9) return "LSP1: typeId out of  
56      scope";  
57  
58      // solhint-disable avoid-tx-origin  
59      if (notifier == tx.origin) revert CannotRegisterEOAsAsAssets(notifier);  
60  
61      bytes32 notifierMapKey = LSP2Utils.generateMappingKey(mapPrefix,  
62      bytes20(notifier));  
63      bytes memory notifierMapValue = IERC725Y(msg.sender).getData(notifierMapKey);
```



```

58
59     if (isReceiving) {
60         // if the amount sent is 0, then do not update the keys
61         uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);
62         if (balance == 0) return "LSP1: balance not updated";
63
64         // if the map value is already set, then do nothing
65         if (bytes12(notifierMapValue) != bytes12(0))
66             return "URD: asset received is already registered";
67
68         (bytes32[] memory receiverDataKeys, bytes[] memory receiverDataValues) =
LSP5Utils
69             .generateReceivedAssetKeys(msg.sender, notifier, notifierMapKey,
interfaceID);
70
71         IERC725Y(msg.sender).setData(receiverDataKeys, receiverDataValues);
72     } else {
73         // if there is no map value for the asset to remove, then do nothing
74         if (bytes12(notifierMapValue) == bytes12(0))
75             return "LSP1: asset sent is not registered";
76         // if it's a token transfer (LSP7/LSP8)
77         uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);
78         if (balance != 0) return "LSP1: full balance is not sent";
79
80         (bytes32[] memory senderDataKeys, bytes[] memory senderDataValues) =
LSP5Utils
81             .generateSentAssetKeys(msg.sender, notifierMapKey, notifierMapValue);
82
83         IERC725Y(msg.sender).setData(senderDataKeys, senderDataValues);
84     }
85 }

```

Recommendation

Change to:

```

59     if (isReceiving) {
60         if (bytes12(notifierMapValue) != bytes12(0))
61             return "URD: asset received is already registered";
62
63         uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);

```



```
64      if (balance == 0) return "LSP1: balance not updated";
```

[WP-N22] LSP-6: Inconsistent

InvalidCompactByteArrayLengthElement check

Issue Description

Unlike in `_verifyAllowedERC725YSingleKey()`, the `length > 32` check is not performed for the error `InvalidCompactByteArrayLengthElement` in `_verifyAllowedERC725YDataKeys()`.

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L599-L678>

```

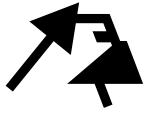
599  function _verifyAllowedERC725YSingleKey(address from, bytes32 inputKey, bytes
      memory allowedERC725YDataKeysCompacted) internal pure {
      @@ 600,626 @@
627      while (pointer < allowedERC725YDataKeysCompacted.length) {
628          /**
629              * save the length of the following allowed key
630              * which is saved in `AllowedERC725YDataKeys[pointer]`
631              */
632          uint256 length =
      uint256(uint8(bytes1(allowedERC725YDataKeysCompacted[pointer])));
633
634          /**
635              * the length of the following key must be under 33 bytes
636              */
637          if (length > 32) revert InvalidCompactByteArrayLengthElement(length);
638
      @@ 639,674 @@
675      }
676
677      revert NotAllowedERC725YDataKey(from, inputKey);
678  }

```

<https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L685-L784>



47



[WP-D23] LSP-6: The corresponding `valueContent` of `valueType` `bytes[CompactByteArray]` should be `bytes[]` instead of `Bytes`

Issue Description

According to the LSP-2-ERC725YJSONSchema.mdvaluecontent document:

`valuecontent`

Describes how to interpret the content of the returned *decoded* value.

Combined with the description of `AddressPermissions:AllowedCalls:<address>` and `AddressPermissions:AllowedERC725YDataKeys:<address>` in LSP-6-KeyManager.md, and their `valueType`, their `valueContent` should be `bytes[]` instead of `Bytes`.

<https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-6-KeyManager.md#implementation>

```
1  [
2    {
3      "name": "AddressPermissions[]",
4      "key":
5        "0xdf30dba06db6a30e65354d9a64c609861f089545ca58c6b4dbe31a5f338cb0e3",
6      "keyType": "Array",
7      "valueType": "address",
8      "valueContent": "Address"
9    },
10   {
11     "name": "AddressPermissions:Permissions:<address>",
12     "key": "0x4b80742de2bf82acb3630000<address>",
13     "keyType": "MappingWithGrouping",
14     "valueType": "bytes32",
15     "valueContent": "BitArray"
16   },
17   {
18     "name": "AddressPermissions:AllowedCalls:<address>",
19     "key": "0x4b80742de2bf393a64c70000<address>",
20     "keyType": "MappingWithGrouping",
21     "valueType": "bytes[CompactByteArray]",
22     "valueContent": "Bytes"
```



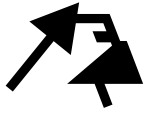
```
22     },
23     {
24         "name": "AddressPermissions:AllowedERC725YDataKeys:<address>",
25         "key": "0x4b80742de2bf866c29110000<address>",
26         "keyType": "MappingWithGrouping",
27         "valueType": "bytes[CompactByteArray]",
28         "valueContent": "Bytes"
29     }
30 ]
```



Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.



Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.