Valory Agent Registries

Smart Contract Audit Report

AUDIT SUMMARY

Valory Agent Registries is a series of smart contracts allowing developers to record off-chain code as Services on-chain.

For this audit, we reviewed the project team's contracts folder at commit v0.3.0 on the team's private GitHub repository.

AUDIT FINDINGS

> *All findings have been resolved, with minimal centralized aspects present.*
> *Date: July 15th, 2022.*
> *Updated: July 19th, 2022 with changes from commit 1a3fbd065a91e7d85544c65476e458b917a22fcd to commit v0.3.0.*
>
> #### Finding #1 - ServiceRegistry - Medium (Resolved)
>
> ***Description:*** *Slashed funds are removed from an Operator's Service balance and not returned when they are slashed. As the Operator's Service balance is decreased, slashed funds will also not be returned when unbonding.*
> ***Risk/Impact:*** *Any slashed funds will remain within the contract.*
> ***Recommendation:*** *The team should return funds as they are slashed or add functionality to allow slashed funds to be withdrawn.*
> ***Resolution:*** *The team has implemented a Drainer address to allow them to withdraw slashed funds.*
>
> ---
>
> #### Finding #2 - UnitRegistry - Informational (Resolved)
>
> ***Description:*** *The subComponentIds.length is used multiple times, including in a loop, when adding a new subComponent Id.*
> ***Recommendation:*** *The team should consider declaring a variable equal to subComponentIds.length to save gas on each use.*
> ***Resolution:*** *The team has implemented the above recommendation.*

CONTRACTS OVERVIEW

> - *The contracts utilize ReentrancyGuard to prevent against reentrancy attacks in applicable functions.*
> - *As the contracts are implemented with Solidity v0.8.15, they are safe from any possible overflows/underflows.*
> - *The team has optimized the contracts' gas usage through variable packing and efficient logic where possible.*
>
> *AgentRegistry & ComponentRegistry:*
> - *These contracts are used to create and manage Components and Agents (units).*
> - *The Manager address may create a new unit for the Unit Owner.*
> - *Each unit is associated with a hash.*
> - *The unit's dependencies, or SubComponents, must be provided in ascending order by unit ID.*
> - *The Unit Owner will be minted an NFT representing ownership of the unit.*
> - *The onERC721Received function must be implemented if the Unit Owner is a contract.*
> - *A Unit Owner may associate additional unused hashes with any of their units.*
> - *The owner may transfer ownership at any time.*

- *The owner may update the Manager address at any time.*
- *The owner may update the base URI at any time.*
- *The contract is ERC721 compliant.*

ServiceRegistry:
- *This contract is used to create and manage Services.*
- *Services may be in one of several states depending on the most recent action performed.*
- *The Manager address may create a new Service for the Service Owner.*
- *The hash associated with the service and the threshold for a successful MultiSig wallet call must be provided in addition to the number of Agents and their corresponding number of "slots" and "bond" cost in ETH.*
- *The threshold must be at least 2/3 of the total number of Agents.*
- *Each Agent's number of slots and bond cost are stored using a 2-part key consisting of the Service ID and the Agent ID.*
- *The Agent IDs associated with each Agent must be provided in ascending order.*
- *A newly created Service will be in the PreRegistration state.*
- *The Service Owner is minted an NFT representing ownership of the Service.*
- *The onERC721Received function must be implemented if the Service Owner is a contract.*
- *The Manager address may update any Services in the PreRegistration state.*
- *An update may remove existing Agents, add new Agents, modify an Agents slots or bond, update the MultiSig threshold, and add a new hash to a Service.*
- *An Agent is removed from the service if the Agent ID is provided with no corresponding slots in the Agent parameters.*
- *Similarly to creating a Service, the Agent IDs must be provided in ascending order when updating.*
- *The Manager address may activate the registration of any Services in the PreRegistration state.*
- *The Service Owner must pay the "security deposit" amount of ETH.*
- *The security deposit is equal to the largest bond of any of the Agents associated with the Service being activated.*
- *The Service will then be in the ActiveRegistration state.*
- *The Manager address may register Agents for any Service in the ActiveRegistration state for an Operator.*
- *Each Agent instance (the Agent's address and Agent ID) will be associated with the Operator for the corresponding Service being registered.*
- *The Agent instances are stored using a 2-part key consisting of the Service ID and the Operator ID.*
- *Services may have multiple Operators, but each Agent instance may only have one Operator.*
- *An Operator may not be an Agent instance.*
- *The Operator must pay the total bond of all Agents they are registering. The bond is stored in the Operator's balance for the Service.*
- *The Service will be in the FinishedRegistration state once all of the Agents for the Service have been registered.*
- *The Manager Address may deploy any Service in the FinishedRegistration state.*
- *Deploying a Service will create a MultiSig wallet from one of the approved implementations.*
- *The Agent instances provided in the creation process will serve as the signers for the MultiSig wallet.*
- *The Service will then be in the Deployed state.*
- *The MultiSig associated with a Service in the Deployed state may "slash" funds.*
- *Slashing will remove a specified amount of funds from the corresponding Agent's Operator's balance.*
- *The Drainer address may withdraw any slashed funds from the contract.*
- *The Manager Address may terminate any Service that is not in the PreRegistration or TerminatedBonded states.*
- *Terminating a Service will delete the Agent instances associated with the Service and refund the Service Owner their security deposit.*
- *The Service will be in the Terminated Bonded state if it had associated Agent Instances.*
- *The Service will be in the PreRegistration state if it did not have associated Agent Instances.*
- *The Manager address may unbond an Operator for any Service in the TerminatedBonded state.*
- *Unbonding will remove the association between the Operator's Agent instances and the Service, and the association with the Operator Service key.*
- *The Operator will be refunded the total bond amount of the unassociated Agent instances and the Operator's Service balance will be reset to 0.*
- *The Service will be in the PreRegistration state if it has no associated Agent instances after unbonding.*
- *The owner may add and remove an address as a valid MultiSig wallet implementation at any time.*

- *The owner may transfer ownership at any time.*
- *The owner may update the Manager and Drainer addresses at any time.*
- *The owner may update the base URI at any time.*
- *The contract is ERC721 compliant.*

RegistriesManager:
- *This contract serves as a wrapper and the Manager address for the ComponentRegistry and AgentRegistry contracts.*
- *Any address may create a new Agent or Component (unit).*
- *The hash associated with the unit and the dependencies must be provided.*
- *The unit owner may add additional hashes to any of their units.*
- *The owner may update ownership at any time.*
- *The owner may pause the contract, preventing the creation of units, at any time.*

ServiceManager:
- *This contract serves as a wrapper and the Manager address for the ServiceRegistry contract.*
- *Any address may use this contract to create a Service.*
- *The user must provide the number of Agents associated with the Service, each Agents corresponding Agent ID, and the required "bond" that must be paid by each Agent.*
- *The Service Owner may update any of their Services while they are in the PreRegistration state.*
- *The Service Owner may activate the registration of any of their Services by providing the security deposit amount of ETH.*
- *A Service Operator may register Agents associated with any Services in the ActiveRegistration state; each Agent requires paying their corresponding bond amount in ETH.*
- *The Service Owner may deploy any of their Services that are in the ActiveRegistration state.*
- *The Service Owner may terminate any of their Services that are not in the PreRegistration or TerminatedBonded state.*
- *A Service Operator may unbond any of their Services that are in the TerminatedUnbonded state.*
- *The owner may update ownership at any time.*
- *The owner may pause the contract, preventing the creation of Services, at any time.*

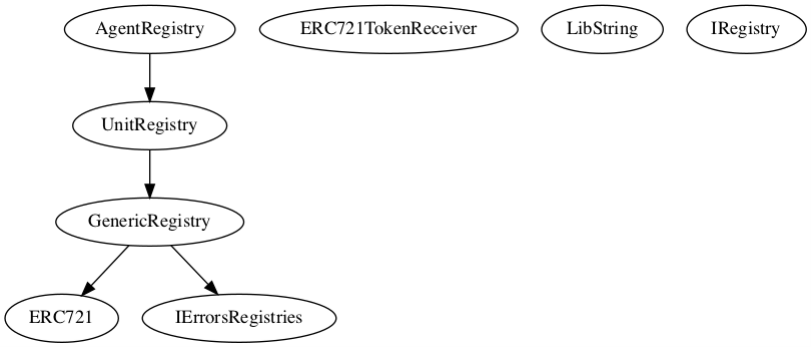
AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	The contracts contain minimal ownership controls.	PASS
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS

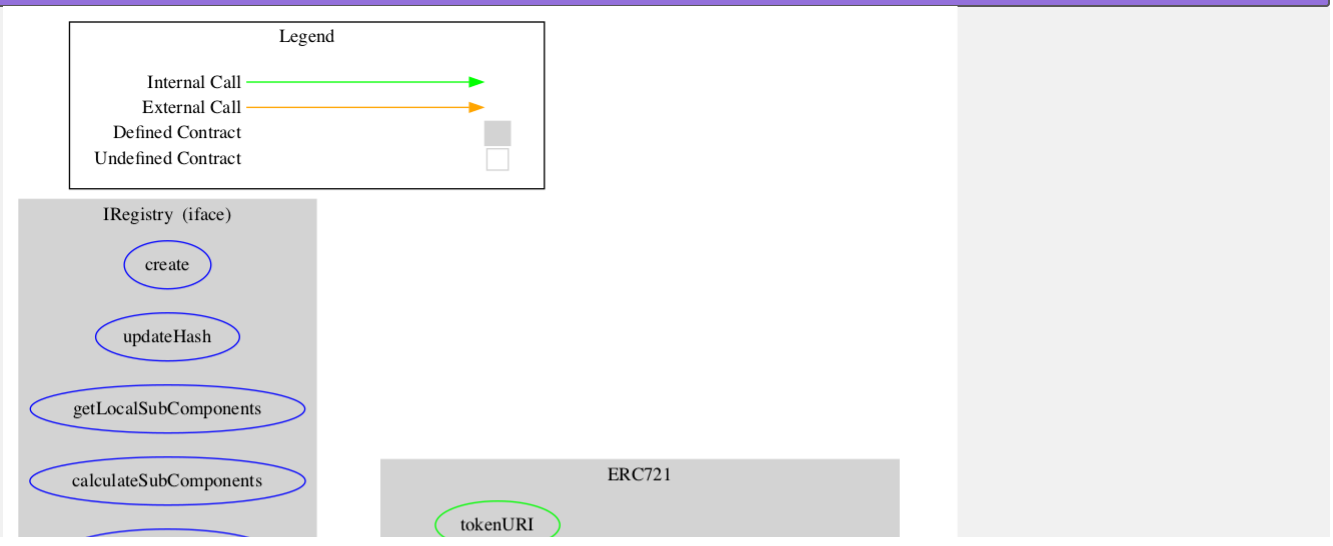
Vulnerability Category	Notes	Result
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

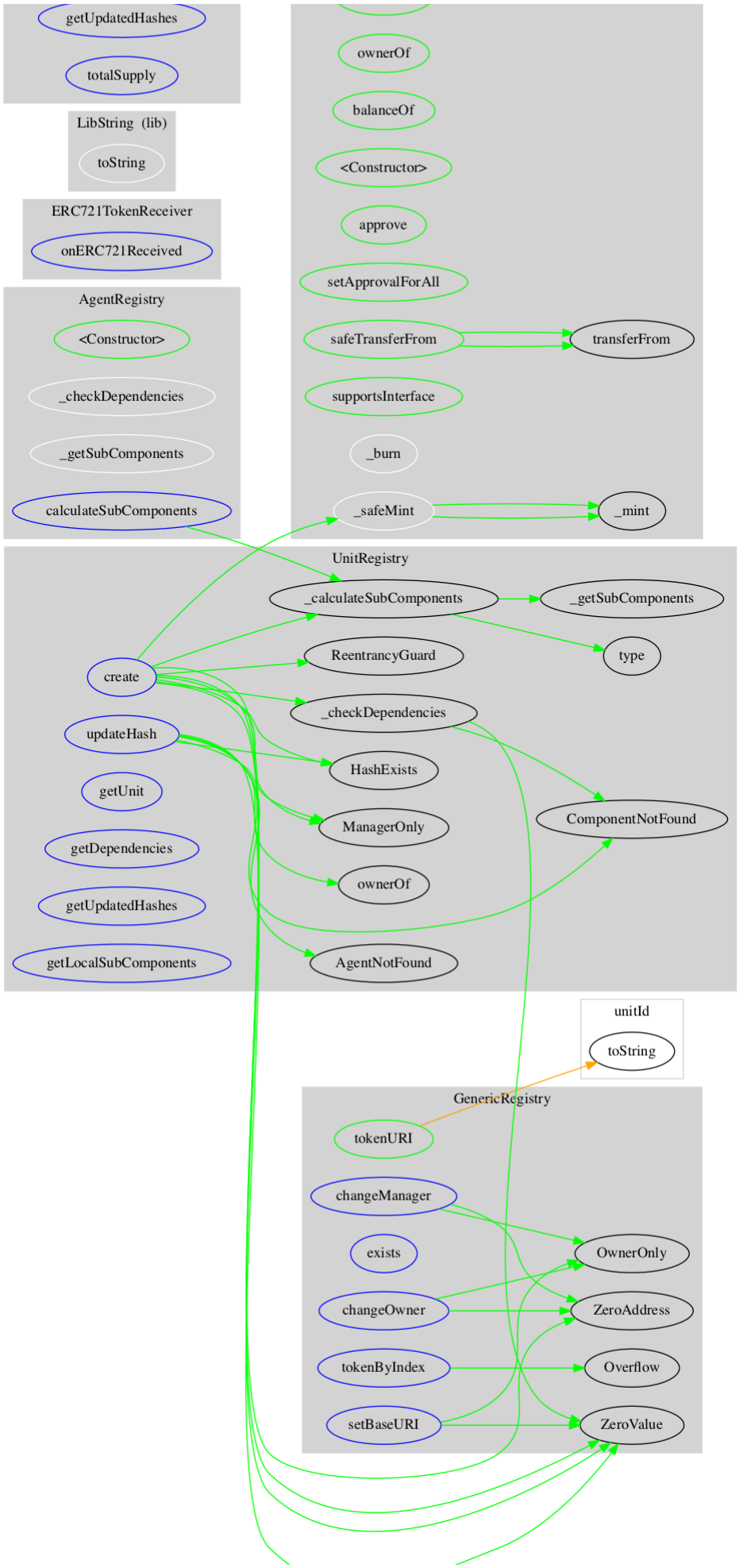
AgentRegistry Contract

INHERITANCE CHART



FUNCTION GRAPH




getUpdatedHashes
totalSupply
LibString (lib)
toString
ERC721TokenReceiver
onERC721Received
AgentRegistry
<Constructor>
_checkDependencies
_getSubComponents
calculateSubComponents
ownerOf
balanceOf
<Constructor>
approve
setApprovalForAll
safeTransferFrom
transferFrom
supportsInterface
_burn
_safeMint
_mint
UnitRegistry
_calculateSubComponents
_getSubComponents
type
create
ReentrancyGuard
_checkDependencies
updateHash
HashExists
getUnit
ManagerOnly
ComponentNotFound
getDependencies
ownerOf
getUpdatedHashes
getLocalSubComponents
AgentNotFound
unitId
toString
GenericRegistry
tokenURI
changeManager
exists
OwnerOnly
changeOwner
ZeroAddress
tokenByIndex
Overflow
setBaseURI
ZeroValue

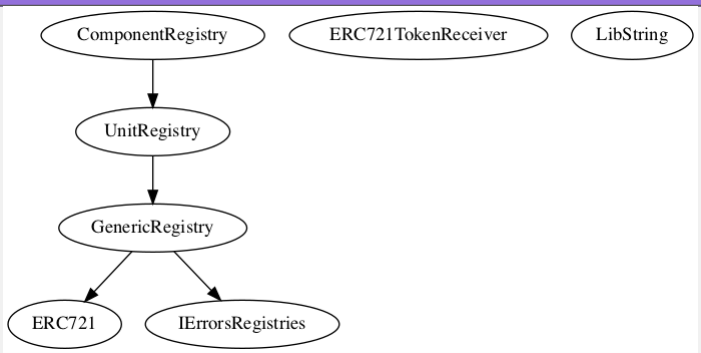
FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  ERC721
   - [Pub] tokenURI
   - [Pub] ownerOf
   - [Pub] balanceOf
   - [Pub] Constructor #
   - [Pub] approve #
   - [Pub] setApprovalForAll #
   - [Pub] transferFrom #
   - [Pub] safeTransferFrom #
   - [Pub] safeTransferFrom #
   - [Pub] supportsInterface
   - [Int] _mint #
   - [Int] _burn #
   - [Int] _safeMint #
   - [Int] _safeMint #

+  ERC721TokenReceiver
   - [Ext] onERC721Received #

+ [Lib] LibString
   - [Int] toString

+ [Int] IErrorsRegistries

+  GenericRegistry (IErrorsRegistries, ERC721)
   - [Ext] changeOwner #
   - [Ext] changeManager #
   - [Ext] exists
   - [Pub] tokenURI
   - [Ext] setBaseURI #
   - [Ext] tokenByIndex

+  UnitRegistry (GenericRegistry)
   - [Int] _checkDependencies #
   - [Ext] create #
   - [Ext] updateHash #
   - [Ext] getUnit
   - [Ext] getDependencies
   - [Ext] getUpdatedHashes
   - [Ext] getLocalSubComponents
   - [Int] _getSubComponents
   - [Int] _calculateSubComponents

+ [Int] IRegistry
   - [Ext] create #
   - [Ext] updateHash #
   - [Ext] getLocalSubComponents
   - [Ext] calculateSubComponents
   - [Ext] getUpdatedHashes
   - [Ext] totalSupply
```

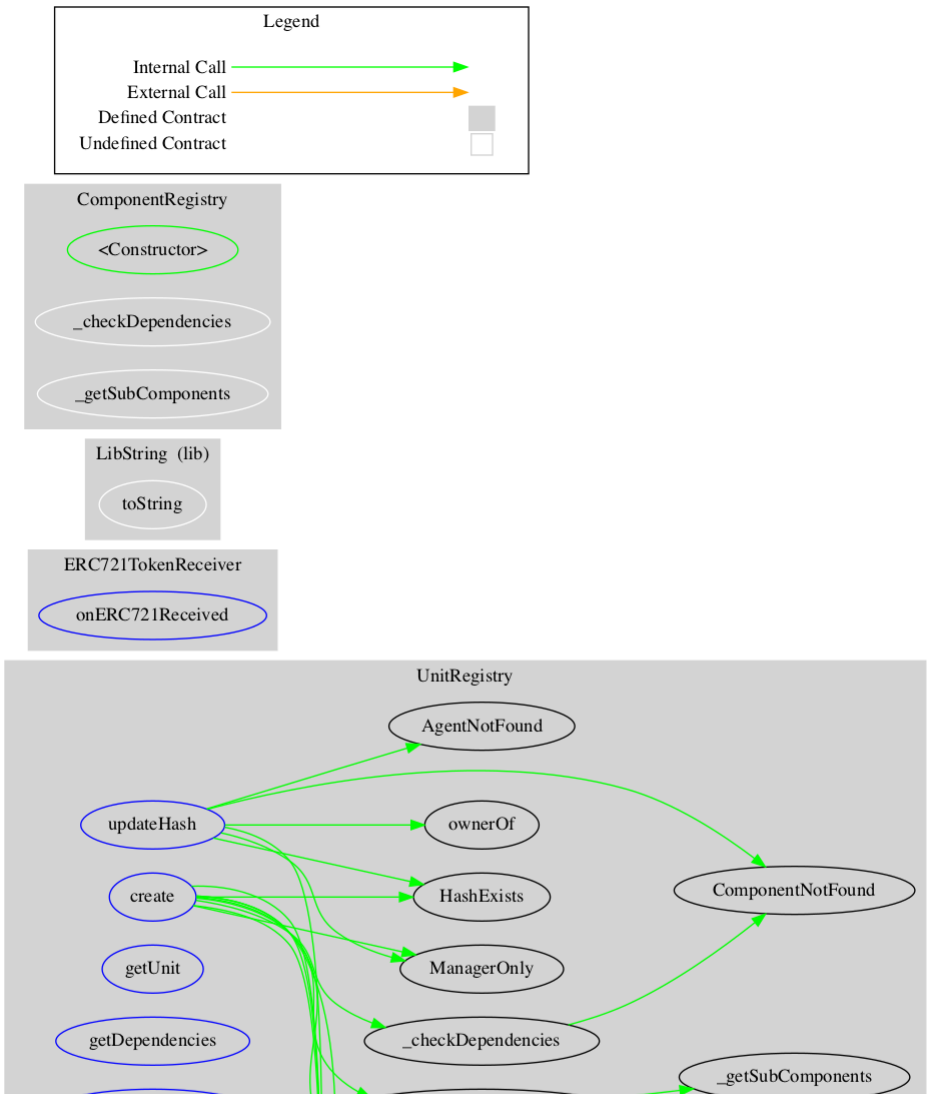
```
+  AgentRegistry (UnitRegistry)
   - [Pub] Constructor #
      - modifiers: ERC721
   - [Int] _checkDependencies #
   - [Int] _getSubComponents
   - [Ext] calculateSubComponents
```

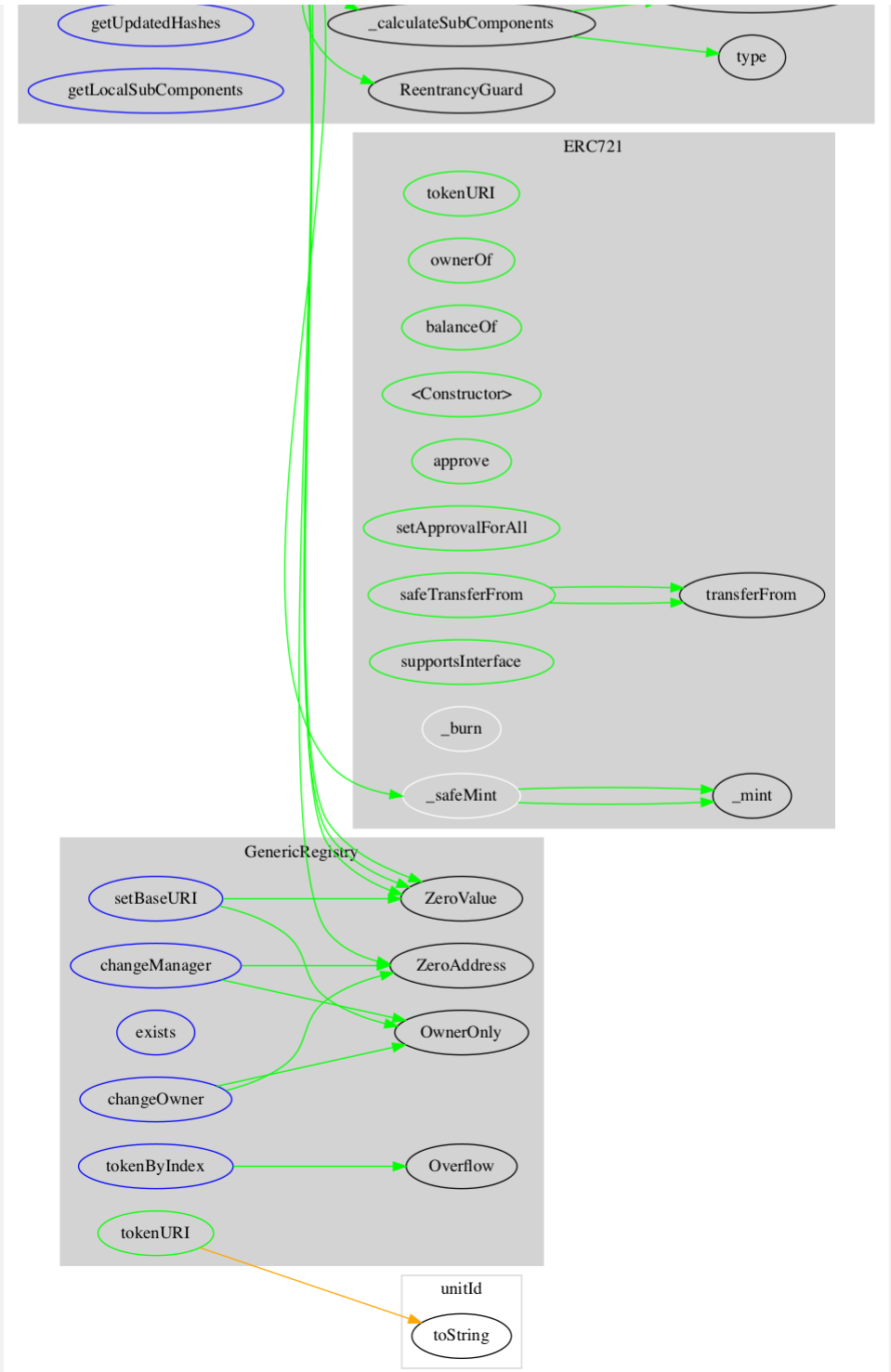
ComponentRegistry Contract

INHERITANCE CHART



FUNCTION GRAPH





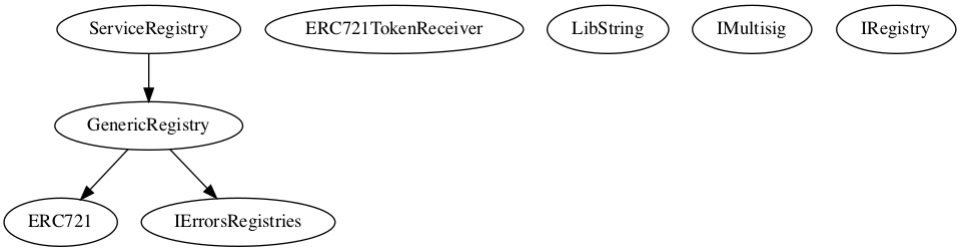
FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  ERC721
- [Pub] tokenURI
- [Pub] ownerOf
```

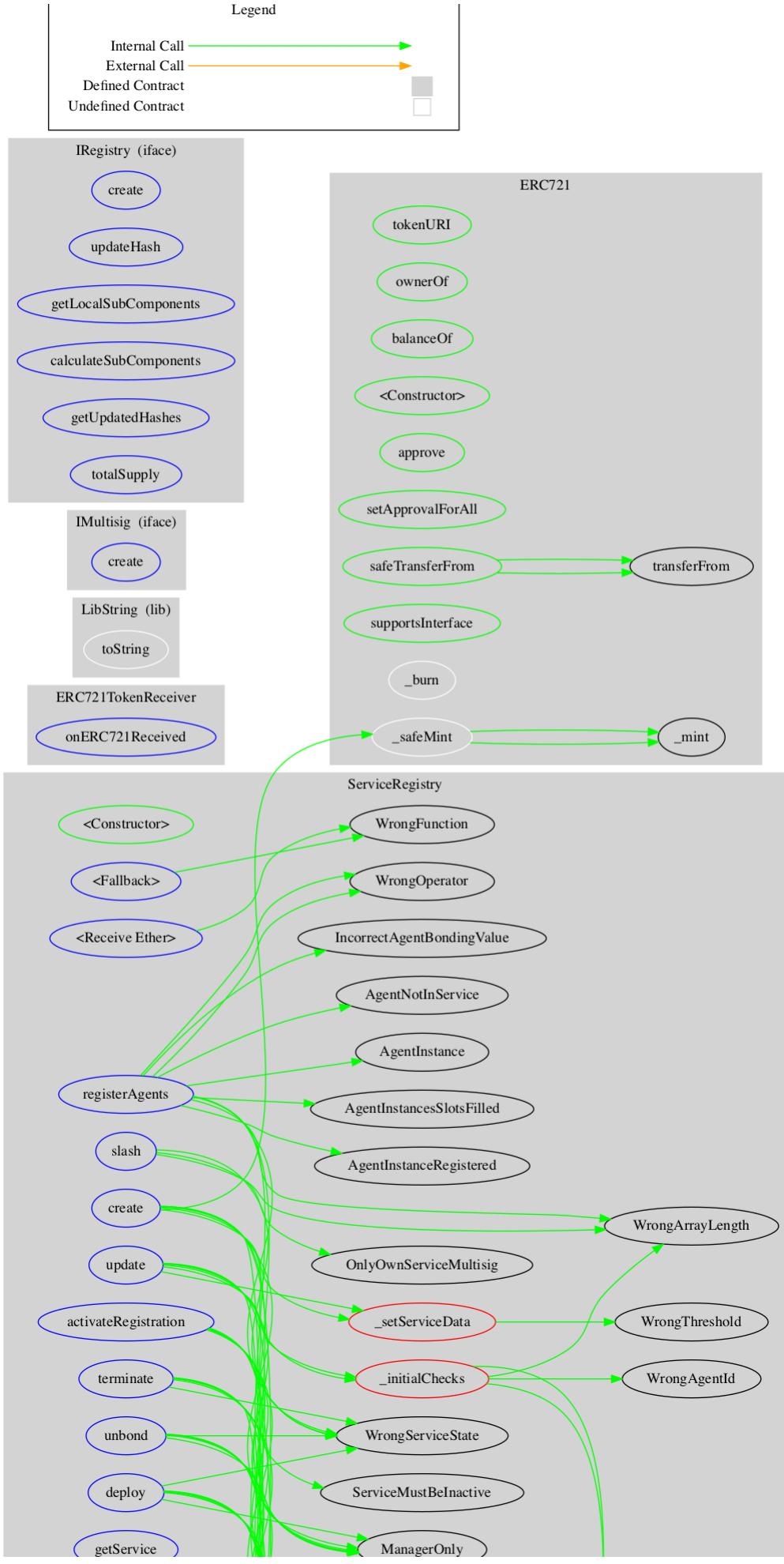
```
 - [Pub] balanceOf
 - [Pub] Constructor #
 - [Pub] approve #
 - [Pub] setApprovalForAll #
 - [Pub] transferFrom #
 - [Pub] safeTransferFrom #
 - [Pub] safeTransferFrom #
 - [Pub] supportsInterface
 - [Int] _mint #
 - [Int] _burn #
 - [Int] _safeMint #
 - [Int] _safeMint #

+  ERC721TokenReceiver
 - [Ext] onERC721Received #

+ [Lib] LibString
 - [Int] toString

+ [Int] IErrorsRegistries

+  GenericRegistry (IErrorsRegistries, ERC721)
 - [Ext] changeOwner #
 - [Ext] changeManager #
 - [Ext] exists
 - [Pub] tokenURI
 - [Ext] setBaseURI #
 - [Ext] tokenByIndex

+  UnitRegistry (GenericRegistry)
 - [Int] _checkDependencies #
 - [Ext] create #
 - [Ext] updateHash #
 - [Ext] getUnit
 - [Ext] getDependencies
 - [Ext] getUpdatedHashes
 - [Ext] getLocalSubComponents
 - [Int] _getSubComponents
 - [Int] _calculateSubComponents

+  ComponentRegistry (UnitRegistry)
 - [Pub] Constructor #
    - modifiers: ERC721
 - [Int] _checkDependencies #
 - [Int] _getSubComponents
```

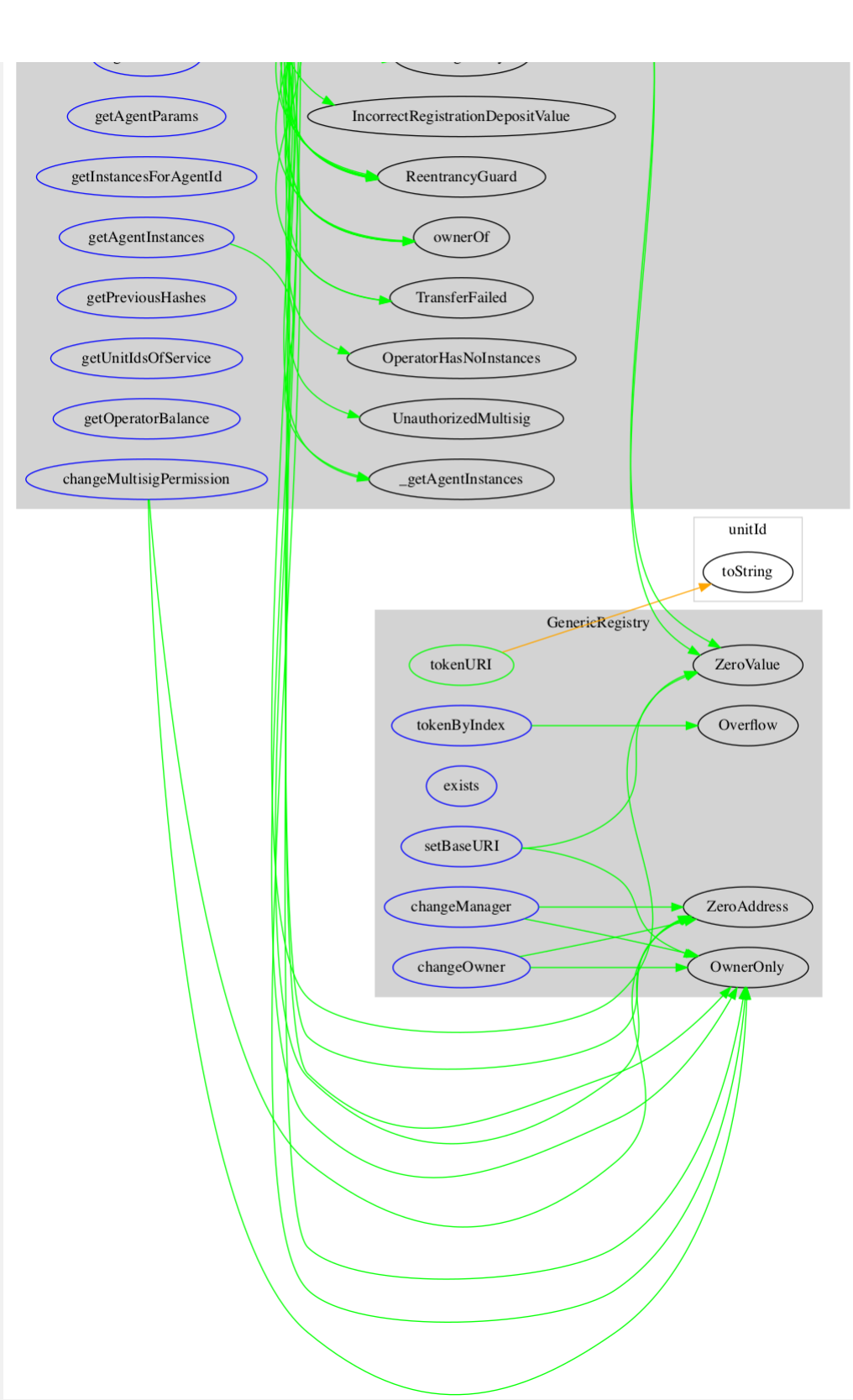
ServiceRegistry Contract

INHERITANCE CHART



FUNCTION GRAPH


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IRegistry (iface)
create
updateHash
getLocalSubComponents
calculateSubComponents
getUpdatedHashes
totalSupply
IMultisig (iface)
create
LibString (lib)
toString
ERC721TokenReceiver
onERC721Received
ERC721
tokenURI
ownerOf
balanceOf
<Constructor>
approve
setApprovalForAll
safeTransferFrom
transferFrom
supportsInterface
_burn
_safeMint
_mint
ServiceRegistry
<Constructor>
<Fallback>
<Receive Ether>
registerAgents
slash
create
update
activateRegistration
terminate
unbond
deploy
getService
WrongFunction
WrongOperator
IncorrectAgentBondingValue
AgentNotInService
AgentInstance
AgentInstancesSlotsFilled
AgentInstanceRegistered
WrongArrayLength
OnlyOwnServiceMultisig
_setServiceData
WrongThreshold
_initialChecks
WrongAgentId
WrongServiceState
ServiceMustBeInactive
ManagerOnly



FUNCTIONS OVERVIEW

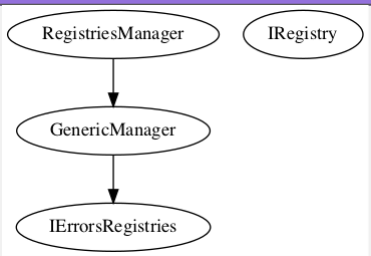
```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public
```

```
 +  ERC721
 - [Pub] tokenURI
 - [Pub] ownerOf
 - [Pub] balanceOf
 - [Pub] Constructor #
 - [Pub] approve #
 - [Pub] setApprovalForAll #
 - [Pub] transferFrom #
 - [Pub] safeTransferFrom #
 - [Pub] safeTransferFrom #
 - [Pub] supportsInterface
 - [Int] _mint #
 - [Int] _burn #
 - [Int] _safeMint #
 - [Int] _safeMint #

+  ERC721TokenReceiver
 - [Ext] onERC721Received #

+ [Lib] LibString
 - [Int] toString

+ [Int] IErrorsRegistries

+  GenericRegistry (IErrorsRegistries, ERC721)
 - [Ext] changeOwner #
 - [Ext] changeManager #
 - [Ext] exists
 - [Pub] tokenURI
 - [Ext] setBaseURI #
 - [Ext] tokenByIndex

+ [Int] IMultisig
 - [Ext] create #

+ [Int] IRegistry
 - [Ext] create #
 - [Ext] updateHash #
 - [Ext] getLocalSubComponents
 - [Ext] calculateSubComponents
 - [Ext] getUpdatedHashes
 - [Ext] totalSupply

+  ServiceRegistry (GenericRegistry)
 - [Pub] Constructor #
    - modifiers: ERC721
 - [Ext] Fallback ($)
 - [Ext] Receive Ether ($)
 - [Prv] _initialChecks
 - [Prv] _setServiceData #
 - [Ext] create #
 - [Ext] update #
 - [Ext] activateRegistration ($)
 - [Ext] registerAgents ($)
 - [Ext] deploy #
 - [Ext] slash #
 - [Ext] terminate #
 - [Ext] unbond #
 - [Ext] getService
 - [Ext] getAgentParams
 - [Ext] getInstancesForAgentId
 - [Prv] _getAgentInstances
 - [Ext] getAgentInstances
 - [Ext] getPreviousHashes
 - [Ext] getUnitIdsOfService
```

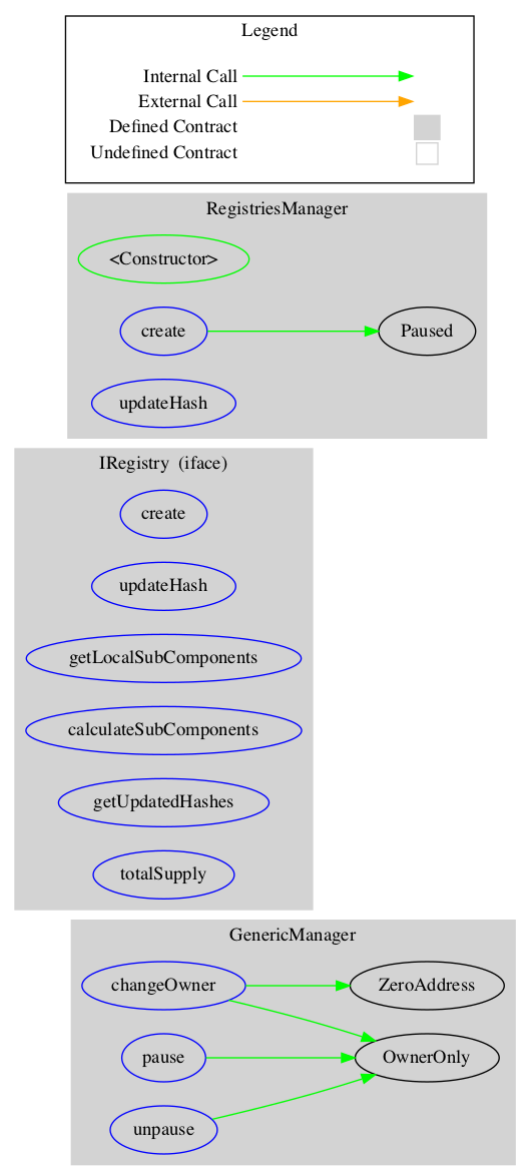
```
- [Ext] getOperatorBalance
- [Ext] changeMultisigPermission #
```

RegistriesManager Contract

INHERITANCE CHART



FUNCTION GRAPH

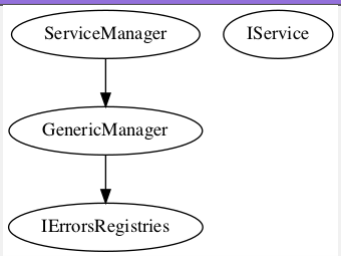


FUNCTIONS OVERVIEW

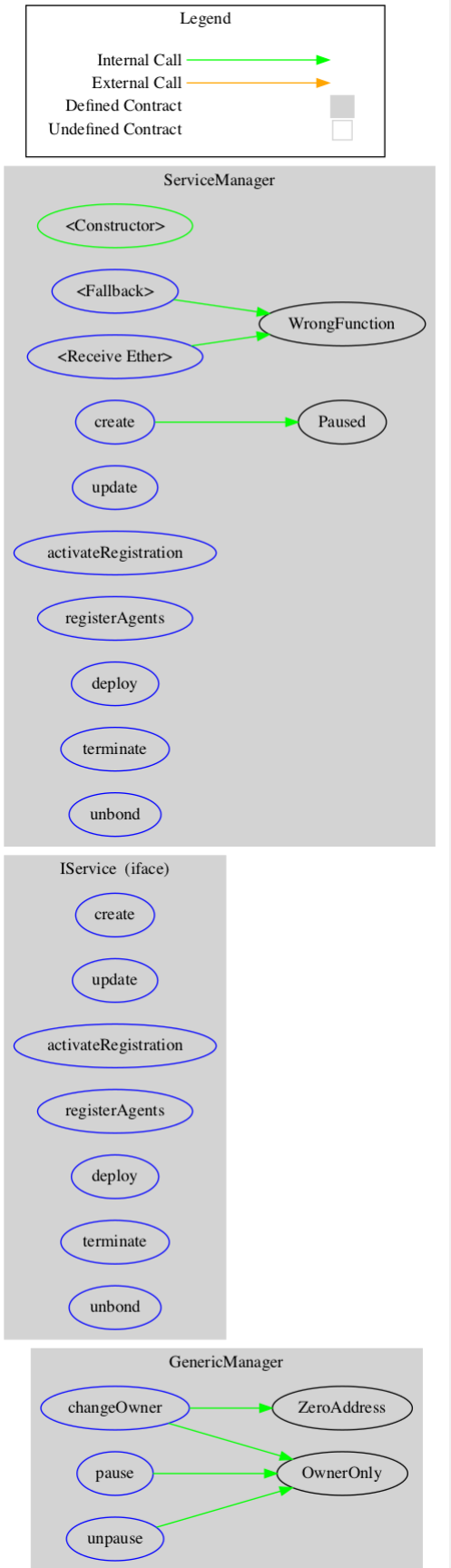
```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+ [Int] IErrorsRegistries

+  GenericManager (IErrorsRegistries)
   - [Ext] changeOwner #
   - [Ext] pause #
   - [Ext] unpause #

+ [Int] IRegistry
   - [Ext] create #
   - [Ext] updateHash #
   - [Ext] getLocalSubComponents
   - [Ext] calculateSubComponents
   - [Ext] getUpdatedHashes
   - [Ext] totalSupply

+  RegistriesManager (GenericManager)
   - [Pub] Constructor #
   - [Ext] create #
   - [Ext] updateHash #
```

ServiceManager Contract

INHERITANCE CHART



FUNCTION GRAPH



FUNCTIONS OVERVIEW

```
($) = payable function
```

```
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+ [Int] IErrorsRegistries

+  GenericManager (IErrorsRegistries)
   - [Ext] changeOwner #
   - [Ext] pause #
   - [Ext] unpause #

+ [Int] IService
   - [Ext] create #
   - [Ext] update #
   - [Ext] activateRegistration ($)
   - [Ext] registerAgents ($)
   - [Ext] deploy #
   - [Ext] terminate #
   - [Ext] unbond #

+  ServiceManager (GenericManager)
   - [Pub] Constructor #
   - [Ext] Fallback ($)
   - [Ext] Receive Ether ($)
   - [Ext] create #
   - [Ext] update #
   - [Ext] activateRegistration ($)
   - [Ext] registerAgents ($)
   - [Ext] deploy #
   - [Ext] terminate #
   - [Ext] unbond #
```

ABOUT SOLIDITY FINANCE

Solidity Finance was founded in 2020 and quickly grew to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1300+ solidity smart contract audits covering all major project types and protocols, securing a total of over $10 billion U.S. dollars in on-chain value.
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.

GO HOME