```
{
  "settings": {
    "evmVersion": "shanghai",
    "libraries": {
      "": {}
    },
    "metadata": {
      "bytecodeHash": "ipfs",
      "useLiteralContent": true
    },
    "optimizer": {
      "enabled": true,
      "runs": 17766
    },
    "remappings": [],
    "viaIR": true,
    "outputSelection": {
      "*": {
        "*": [
          "evm.bytecode.object",
          "evm.deployedBytecode.object",
          "evm.deployedBytecode.immutableReferences",
          "metadata"
        ]
      }
    }
  },
  "sources": {
    "@nfts2me/contracts/important/README.sol": {
      "content": "/** ---------------------------------------------------
------------------------- //\n *
//\n *                 Smart contract generated by https://nfts2me.com
//\n *
//\n *                                       .::.
//\n *                                     ......
//\n *                                 ....        ::.
//\n *                              .:..           :: ...
//\n *                          ..:.               ::     ...
//\n *                        ::.      ..:--       ::.       ...
//\n *                       .:    ..:::::-==:     :::::..     :
//\n *                       .:    :::::::-====:   ::::::::    :
//\n *                       .:    :::::::-======. ::::::::    :
//\n *                       .:    :::::::-=======-::::::::    :
//\n *                       .:    :::::::-========-:::::::    :
//\n *                       .:    ::::::::========-:::::::    :
//\n *                       .:    :::::::. .======-:::::::    :
//\n *                       .:    :::::::.   :====-:::::::    :
//\n *                       .:     .:::::.     -==-:::::.     :
//\n *                        .:.       .:.      .--:..      ...
//\n *                           .:.      :.               ...
//\n *                              .... :.           ....
//\n *                                  .:.        .:.
//\n *                                       .::::.
//\n *                                       :--.
```

```
//\n *
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/             //\n *   More info at:
https://docs.nfts2me.com/                                     //\n * ----
-------------------------------------------------------------------------
*/\n\n/// SPDX-License-Identifier: UNLICENSED\npragma solidity
^0.8.17;\n\n/// @title NFTs2Me.com Smart Contracts README\n/// @author
The NFTs2Me Team\n/// @notice Read our terms of service\n///
@custom:security-contact security@nfts2me.com\n/// @custom:terms-of-
service https://nfts2me.com/terms-of-service/\n/// @custom:website
https://nfts2me.com/\ninterface Readme {\n    function n2mVersion()
external pure returns (uint256);\n }\n"
    },
    "@nfts2me/contracts/interfaces/IN2MCrossFactory.sol": {
      "content": "/** -----------------------------------------------------
------------------------- //\n *
//\n *                                        .:::.
//\n *                                     .:::::::.
//\n *                                     ::::::::.
//\n *                                  .:::::::::.
//\n *                              ..:::.             ..
//\n *                           .::::.              ::::..
//\n *                       ..:::..                 ::::::::.
//\n *                    .::::.                     :::.  ..:::.
//\n *                ..::::..                       :::.      .:::.
//\n *             .::::.                            :::.
.:::..           //\n *        .:::..                    ..
:::.            .::::.        //\n *     .::::.               ..:::=-
::::               ..:::.     //\n *    :::.             .::::::===:
::::::.              .::::   //\n *   .::.         .:::::::::::=====.
::::::::::.            .::.  //\n *   .::
.:::::::::::::::=======.          :::::::::::::..          ::.  //\n *
.::        .::::::::::::::::========-        ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==========:
::::::::::::::::         ::.  //\n *   .::
.::::::::::::::::============:     ::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::===============-.
::::::::::::::::         ::.  //\n *   .::
.::::::::::::::::=================::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::=================-
:::::::::::::::         ::.  //\n *   .::
.::::::::::::::::==================-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-===============-
:::::::::::::::         ::.  //\n *   .::        .::::::::::::::::
.==============-:::::::::::::::        ::.  //\n *   .::
```

```
.::::::::::::::::      :============-:::::::::::::::         ::.  //\n *
.::         .::::::::::::::::        :==========-:::::::::::::::
::.  //\n *   .::         .::::::::::::::::         .-========-
:::::::::::::::         ::.  //\n *   .::          .::::::::::::::
.=======-:::::::::::::.         ::.  //\n *   .::.
.::::::::::            .=====-::::::::::..           ::.  //\n *
:::..              ..:::::::             :===-::::::..              .:::.
//\n *      .:::..              .:::               -=-:::.
.::::.    //\n *         .::::.           .:::                  ..
.::::.       //\n *           .::::.         .:::
..:::.           //\n *               .:::.      .:::
.::::.            //\n *                 .::::..  .:::
..:::..               //\n *                   .::::.:::
.::::.                  //\n *                   ..::::
..:::..                     //\n *                          .:
.::::.                        //\n *
:::::.::::.                          //\n *
::::::::.                             //\n *
::::::::.                             //\n *
.::::.                               //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/             //\n * ---------------
--------------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity
^0.8.17;\n\ninterface IN2MCrossFactory {\n    function
getN2MTreasuryAddress() external pure returns (address);\n    function
ownerOf(uint256 tokenId) external view returns (address);\n    function
strictOwnerOf(uint256 tokenId) external view returns (address);\n}\n"7
    },
    "@nfts2me/contracts/interfaces/IN2M_ERCBase.sol": {
      "content": "/** ---------------------------------------------------
------------------------- //\n *
//\n *                                    .:::.
//\n *                                 .::::::::.
//\n *                                 ::::::::.
//\n *                              .::::::::::.
//\n *                          ..:::.            ..
//\n *                       .::::.               ::::..
//\n *                   ..:::..                  ::::::::.
//\n *                .::::.                      :::.  ..:::.
//\n *            ..:::..                          :::.      .:::.
//\n *         .:::::.                             :::.
.:::..           //\n *          .:::..              ..
:::.            .::::.        //\n *     .::::.              ..:::=-
::::              ..::::.     //\n *    :::.             .::::::::===:
::::::.              .::::   //\n *   .::.            .::::::::::::=====.
```

```
::::::::::.              .::.  //\n *   .::
.:::::::::::::::=======.          :::::::::::::..          ::.  //\n *
.::        .::::::::::::::::=========-         ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==========:
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     ::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::===============-.
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
:::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-===============-
:::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.==============-:::::::::::::::        ::.  //\n *   .::
.::::::::::::::::     :============-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::      :==========-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-========-
:::::::::::::::        ::.  //\n *   .::         .::::::::::::::
.=======-:::::::::::::.         ::.  //\n *   .::.
.::::::::::           .=====-::::::::::..           ::.  //\n *
:::..             ..:::::::             :===-::::::..             .:::.
//\n *      .::::..              .:::              -=-:::.
.::::.    //\n *         .::::.           .:::                ..
.::::.       //\n *           .::::.         .:::
..:::.           //\n *                .:::.      .:::
.::::.              //\n *                 .::::..  .:::
..:::..                //\n *                .:::::.:::
.::::.                  //\n *                  ..::::
..:::..                    //\n *                        .:
.::::.                      //\n *
:::::.::::.                         //\n *
:::::::::.                           //\n *
::::::::.                            //\n *
.::::.                              //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/            //\n * ----------------
------------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.17;\n\nimport
\"@openzeppelin/contracts-
upgradeable/interfaces/IERC2981Upgradeable.sol\";\nimport
\"./IN2M_ERCStorage.sol\";\n\ninterface IN2M_ERCBase is
IERC2981Upgradeable, IN2M_ERCStorage {\n    /// @notice To be called to
```

```
create the collection. Can only be called once.\n    function
initialize\n    (\n        string memory tokenName,\n        string
memory tokenSymbol,\n        uint256 iMintPrice,\n        bytes32
baseURICIDHash,\n        bytes32 placeholderImageCIDHash,\n
RevenueAddress[] calldata revenueAddresses,\n        address
iErc20PaymentAddress,\n        uint32 iTotalSupply,\n        uint16
iRoyaltyFee,\n        bool soulboundCollection,\n        MintingType
iMintingType\n    ) external payable;\n\n    /// @notice A descriptive
name for a collection of NFTs in this contract\n    function name()
external view returns (string memory);\n\n    /// @notice An abbreviated
name for NFTs in this contract\n    /// @return the collection symbol\n
function symbol() external view returns (string memory);\n\n    ///
@notice A distinct Uniform Resource Identifier (URI) for a given asset.\n
/// @dev Throws if `_tokenId` is not a valid NFT. URIs are defined in
RFC\n    ///  3986. The URI may point to a JSON file that conforms to the
\"ERC721\n    ///  Metadata JSON Schema\".\n    function tokenURI(uint256
tokenId) external view returns (string memory);\n\n    /// @notice Mints
one NFT to the caller (msg.sender). Requires `minting type` to be
`sequential` and the `mintPrice` to be send (if `Native payment`) or
approved (if `ERC-20` payment).\n    function mint() external
payable;\n\n    /// @notice Mints `amount` NFTs to the caller
(msg.sender). Requires `minting type` to be `sequential` and the
`mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment).\n    /// @param amount The number of NFTs to mint\n    function
mint(uint256 amount) external payable;\n\n    /// @notice Mints `amount`
NFTs to the caller (msg.sender) with a given `affiliate`. Requires
`minting type` to be `sequential` and the `mintPrice` to be send (if
`Native payment`) or approved (if `ERC-20` payment).\n    /// @param
amount The number of NFTs to mint\n    /// @param affiliate The affiliate
address\n    function mint(uint256 amount, address affiliate) external
payable;\n\n    /// @notice Mints `amount` NFTs to `to` address. Requires
`minting type` to be `sequential` and the `mintPrice` to be send (if
`Native payment`) or approved (if `ERC-20` payment).\n    /// @param to
The address of the NFTs receiver\n    /// @param amount The number of
NFTs to mint    \n    function mintTo(address to, uint256 amount)
external payable;\n\n    /// @notice Mints `amount` NFTs to `to` address
with a given `affiliate`. Requires `minting type` to be `sequential` and
the `mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment).\n    /// @param to The address of the NFTs receiver\n    ///
@param amount The number of NFTs to mint    \n    /// @param affiliate
The affiliate address\n    function mintTo(address to, uint256 amount,
address affiliate) external payable;\n\n    /// @notice Two phases on-
chain random minting. Mints `amount` NFTs tickets to `to` address.
Requires `minting type` to be `random` and the `mintPrice` to be send (if
`Native payment`) or approved (if `ERC-20` payment). Once minted, those
tickets must be redeemed for an actual token calling `redeemRandom()`.\n
/// @param to The address of the NFTs receiver\n    /// @param amount The
number of NFTs to mint    \n    function mintRandomTo(address to, uint256
amount) external payable;    \n\n    /// @notice Two phases on-chain
random minting. Mints `amount` NFTs tickets to `to` address with a given
`affiliate`. Requires `minting type` to be `random` and the `mintPrice`
to be send (if `Native payment`) or approved (if `ERC-20` payment). Once
minted, those tickets must be redeemed for an actual token calling
`redeemRandom()`.\n    /// @param to The address of the NFTs receiver\n
```

```
/// @param amount The number of NFTs to mint    \n    /// @param
affiliate The affiliate address\n    function mintRandomTo(address to,
uint256 amount, address affiliate) external payable;\n\n    /// @notice
Redeems remaining random tickets generated from `msg.sender` by calling
`mintRandomTo` for actual NFTs.\n    function redeemRandom() external
payable;\n\n    /// @notice Mints `amount` NFTs to `to` address. Requires
`minting type` to be `specify` and the `mintPrice` to be send (if `Native
payment`) or approved (if `ERC-20` payment).\n    /// @param to The
address of the NFTs receiver\n    /// @param tokenIds An array of the
specified tokens. They must not be minted, otherwise, it will revert.\n
function mintSpecifyTo(address to, uint256[] memory tokenIds) external
payable; \n\n    /// @notice Mints `amount` NFTs to `to` address with a
given `affiliate`. Requires `minting type` to be `specify` and the
`mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment).\n    /// @param to The address of the NFTs receiver\n    ///
@param tokenIds An array of the specified tokens. They must not be
minted, otherwise, it will revert.\n    /// @param affiliate The
affiliate address\n    function mintSpecifyTo(address to, uint256[]
memory tokenIds, address affiliate) external payable; \n\n    /// @notice
Mints one NFT to `to` address. Requires `minting type` to be
`customURI`.\n    /// @param to The address of the NFTs receiver\n    ///
@param customURICIDHash The CID of the given token.\n    /// @param
soulbound True if the NFT is a Soulbound Token (SBT). If set, it can't be
transferred.\n    function mintCustomURITo(address to, bytes32
customURICIDHash, bool soulbound) external payable;\n\n    /// @notice
Only owner can call this function. Free of charge. Mints sizeof(`to`) to
`to` addresses. Requires `minting type` to be `sequential`.\n    ///
@param to The addresses of the NFTs receivers\n    /// @param soulbound
True if the NFT is a Soulbound Token (SBT). If set, it can't be
transferred.\n    function airdropSequential(address[] memory to, bool
soulbound) external payable;\n\n    /// @notice Only owner can call this
function. Free of charge. Mints sizeof(`to`) to `to` addresses with
random tokenIds. Requires `minting type` to be `random`.\n    /// @param
to The addresses of the NFTs receivers\n    /// @param soulbound True if
the NFT is a Soulbound Token (SBT). If set, it can't be transferred.\n
function airdropRandom(address[] memory to, bool soulbound) external
payable;\n\n    /// @notice Only owner can call this function. Free of
charge. Mints sizeof(`to`) to `to` addresses with specified tokenIds.
Requires `minting type` to be `specify`.\n    /// @param to The addresses
of the NFTs receivers\n    /// @param tokenIds An array of the specified
tokens. They must not be minted, otherwise, it will revert.\n    ///
@param soulbound True if the NFT is a Soulbound Token (SBT). If set, it
can't be transferred.\n    function airdropSpecify(address[] memory to,
uint256[] memory tokenIds, bool soulbound) external payable;\n\n    ///
@notice Mints `amount` of NFTs to `to` address with optional specified
tokenIds. This function must be called only if a valid `signature` is
given during a whitelisting/presale.\n    /// @param to The addresses of
the NFTs receivers\n    /// @param tokenIds An optional array of the
specified tokens. They must not be minted, otherwise, it will revert.
Only used if minting type is `specify`.\n    /// @param freeMinting True
is minting is free\n    /// @param customFee Zero is fee is different
from `mintingPrice`.\n    /// @param maxAmount Max Amount to be minted
with the given `signature`.\n    /// @param amount Amount to mint.\n
/// @param soulbound True if the NFT is a Soulbound Token (SBT). If set,
it can't be transferred.\n    /// @param signature Valid `signature` for
the presale/whitelist.\n    function mintPresale (\n        address to,
\n        uint256[] memory tokenIds,\n        bool freeMinting, \n
uint256 customFee, \n        uint256 maxAmount,\n        uint256 amount,
\n        bool soulbound,\n        bytes calldata signature) payable
external;\n\n    /// @notice Returns the minting price of one NFT.\n
/// @return Mint price for one NFT in native coin or ERC-20.\n
function mintPrice() external view returns (uint256);\n\n    /// @notice
Returns the current total supply.\n    /// @return Current total
supply.\n    function totalSupply() external view returns (uint256);\n\n
/// @notice Max amount of NFTs to be hold per address.\n    /// @return
Max per address allowed.\n    function maxPerAddress() external view
returns (uint16);\n\n}\n\n"
    },
    "@nfts2me/contracts/interfaces/IN2M_ERCStorage.sol": {
      "content": "/** ---------------------------------------------------
------------------------- //\n *
//\n *                                      .:::.
//\n *                                   .::::::::.
//\n *                                   ::::::::.
//\n *                                .::::::::::.
//\n *                            ..:::.              ..
//\n *                         .::::.                 ::::..
//\n *                     ..::::..                   ::::::::.
//\n *                  .::::.                        :::.  ..:::.
//\n *              ..::::..                          :::.      .:::.
//\n *           .::::.                               :::.
.:::..           //\n *        .:::..                    ..
:::.            .::::.        //\n *     .::::.               ..:::=-
::::               ..:::.     //\n *    :::.               .:::::::===:
::::::.               .::::   //\n *   .::.           .::::::::::::=====.
::::::::::.             .::.  //\n *   .::
.::::::::::::::=======.          ::::::::::::..           ::.  //\n *
.::         .::::::::::::::::========-         ::::::::::::::::
::.  //\n *   .::         .::::::::::::::::==========:
::::::::::::::::         ::.  //\n *   .::
.::::::::::::::::============:     ::::::::::::::::         ::.  //\n *
.::         .::::::::::::::::==============.   ::::::::::::::::
::.  //\n *   .::         .::::::::::::::::===============-.
::::::::::::::::         ::.  //\n *   .::
.::::::::::::::::================::::::::::::::::         ::.  //\n *
.::         .::::::::::::::::=================-::::::::::::::
::.  //\n *   .::         .::::::::::::::::=================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================-::::::::::::::::         ::.  //\n *
.::         .::::::::::::::::================-::::::::::::::
::.  //\n *   .::         .:::::::::::::::: .-===============-
::::::::::::::::        ::.  //\n *   .::         .::::::::::::::
.=============-:::::::::::::::         ::.  //\n *   .::
.::::::::::::::::      :============-::::::::::::::         ::.  //\n *
.::         .::::::::::::::::       :==========-:::::::::::::::
::.  //\n *   .::         .::::::::::::::::         .-========-
::::::::::::::::        ::.  //\n *   .::          .::::::::::::::
.=======-::::::::::::.          ::.  //\n *   .::.
```

```
.::::::::::           .=====-::::::::::..           ::.  //\n *
::::..             ..:::::::              :===-:::::::..             .::::.
//\n *      .::::..             .:::              -==-::::.
.:::::.    //\n *         .:::::.           .:::                 ..
.:::::.       //\n *           .:::::.        .:::
..::::.          //\n *               .::::.      .:::
.:::::.             //\n *                 .:::::..  .:::
..::::..               //\n *                   .:::::.:::
.:::::.                   //\n *                     ..::::
..::::..                      //\n *                         .:
.:::::.                         //\n *
:::::.::::.                        //\n *
::::::::.                           //\n *
:::::::.                            //\n *
.:::::.                            //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/            //\n * ----------------
--------------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.17;\n\nimport
\"../important/README.sol\";\n\ninterface IN2M_ERCStorage is Readme {\n
/// @notice This event is emitted when a token is minted using an
affiliate\n    /// @param affiliate The affiliate address\n    event
AffiliateSell(address indexed affiliate);\n\n    /// @notice Error thrown
when trying to mint a token with a given id which is already minted\n
error TokenAlreadyMinted();\n\n    /// @notice Error thrown when input
variable differ in length\n    error InvalidInputSizesDontMatch();\n\n
/// @notice Error thrown when input variable differ in length\n    error
InputSizeMismatch();\n\n    /// @notice Error thrown when trying to mint
a token with a given invalid id\n    error InvalidTokenId();\n\n    ///
@notice Error thrown when trying to redeem random tickets with no amount
to redeem\n    error NothingToRedeem();\n\n    /// @notice Error thrown
when trying to redeem random tickets too soon\n    error
CantRevealYetWaitABitToBeAbleToRedeem();\n\n    /// @notice Error thrown
when the input amount is not valid\n    error InvalidAmount();\n\n    ///
@notice Error thrown when trying to mint a sold out collection or the
amount to mint exceeds the remaining supply\n    error
CollectionSoldOut();\n\n    /// @notice Error thrown when trying to
presale/whitelist mint and the collection current phase is `closed`\n
error PresaleNotOpen();\n\n    /// @notice Error thrown when trying to
mint and the collection current phase is not `open`\n    error
PublicSaleNotOpen();\n\n    /// @notice Error thrown when trying to mint
but the sale has already finished\n    error SaleFinished();\n\n    ///
@notice Error thrown when trying to mint more than the allowance to
mint\n    error NotEnoughAmountToMint();\n\n    /// @notice Error thrown
when sending funds to a free minting\n    error
InvalidMintFeeForFreeMinting();\n\n    /// @notice Error thrown when the
sent amount is not valid\n    error InvalidMintFee();\n\n    /// @notice
Royalty fee can't be higher than 10%\n    error RoyaltyFeeTooHigh();\n\n
/// @notice Invalid input. Total supply must be greater than zero\n
error TotalSupplyMustBeGreaterThanZero();\n\n    /// @notice Can't set
BaseURI and Placeholder at the same time\n    error
CantSetBaseURIAndPlaceholderAtTheSameTime();\n\n    /// @notice No
BaseURI nor Placeholder set\n    error NoBaseURINorPlaceholderSet();\n\n
/// @notice Can't transfer a Soulbound Token (SBT)\n    error
NonTransferrableSoulboundNFT();\n\n    /// @notice The input revenue
percentages are not valid\n    error InvalidRevenuePercentage();\n\n
/// @notice Can't mint until specified drop date\n    error
WaitUntilDropDate();\n\n    /// @notice Trying to use mintPresale method
in a collection with a minting type that doesn't support whitelist\n
error PresaleInvalidMintingType();\n\n    /// @notice Metadata is already
fixed. Can't change metadata once fixed\n    error
MetadataAlreadyFixed();\n\n    /// @notice Invalid collection minting
type for the current minting function\n    error
InvalidMintingType();\n\n    /// @notice The address exceeded the max per
address amount\n    error MaxPerAddressExceeded();\n\n    /// @notice The
given signature doesn't match the input values\n    error
SignatureMismatch();\n\n    /// @notice Reentrancy Guard protection\n
error ReentrancyGuard();\n\n    /// @notice New Placeholder can't be
empty\n    error NewPlaceholderCantBeEmpty();\n\n    /// @notice New
BaseURI can't be empty\n    error NewBaseURICantBeEmpty();    \n\n    ///
@notice Invalid percentage or discount values\n    error
InvalidPercentageOrDiscountValues();\n\n    /// @notice Can't lower
current percentages\n    error CantLowerCurrentPercentages();\n\n    ///
@notice Contract MetadataURI already fixed\n    error
ContractMetadataURIAlreadyFixed();\n\n    /// @notice Only owner of N2M
can call this function\n    error OnlyOwnerOrN2M();\n\n    /// @notice
Only the given affiliate or N2M can call this function\n    error
OnlyAffiliateOrN2M();\n\n    /// @notice The signature has expired\n
error SignatureExpired();\n\n    /// @notice Invalid phase can't be set
without giving a date, use the proper functions\n    error
InvalidPhaseWithoutDate();\n\n    /// @notice Invalid drop date\n
error InvalidDropDate();\n\n    /// @notice Operator address is
filtered\n    error AddressFiltered(address filtered);\n\n    struct
RandomTicket {\n        uint256 amount;\n        uint256
blockNumberToReveal;\n    }\n\n    struct RevenueAddress {\n
address to;\n        uint16 percentage;\n    }\n\n    struct
AffiliateInformation {\n        bool enabled;\n        uint16
affiliatePercentage;\n        uint16 userDiscount;\n    }\n\n    enum
SalePhase { \n        CLOSED,\n        PRESALE,\n        PUBLIC,\n
DROP_DATE,\n        DROP_AND_END_DATE\n    }\n\n    enum MintingType { \n
SEQUENTIAL, \n        RANDOM, \n        SPECIFY, \n        CUSTOM_URI \n
}\n\n    enum OperatorFilterStatus { \n        ENABLED_NOT_INITIALIZED,
\n        ENABLED_EXISTS,          \n        DISABLED_NOT_INITIALIZED,\n
DISABLED_EXISTS          \n    }\n\n    /// @notice Returns true if the
metadata is fixed and immutable. If the metadata hasn't been fixed yet it
will return false. Once fixed, it can't be changed by anyone.\n
function isMetadataFixed() external view returns (bool);\n\n}\n\n"
    },
    "@nfts2me/contracts/interfaces/IOperatorFilterRegistry.sol": {
      "content": "/** -----------------------------------------------------
------------------------- //\n *
//\n *                                      .:::.
//\n *                                   .:::::::.
//\n *                                   ::::::::.
//\n *                                .::::::::::.
//\n *                            ..:::.             ..
//\n *                         .::::.                ::::..
//\n *                     ..::::..                  ::::::::.
//\n *                  .::::.                       :::.  ..:::.
//\n *              ..:::..                          :::.      .:::.
//\n *           .::::.                              :::.
.:::..           //\n *         .:::..                     ..
:::.            .::::.        //\n *     .::::.                ..:::=-
::::               ..:::.     //\n *    :::.                .:::::::===:
::::::.               .::::   //\n *   .::.            .:::::::::::=====.
::::::::::.             .::.  //\n *   .::
.::::::::::::::=======.           :::::::::::::..           ::.  //\n *
.::        .:::::::::::::::::========-          ::::::::::::::::
::.  //\n *   .::        .:::::::::::::::::==========:
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     :::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::================-.
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================:::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==================-::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::================-::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-================-
::::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.==============-:::::::::::::::        ::.  //\n *   .::
.::::::::::::::::      :============-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::       :===========-::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-========-
::::::::::::::::        ::.  //\n *   .::         .::::::::::::::
.=======-::::::::::::::.         ::.  //\n *   .::.
.::::::::::           .=====-::::::::::..           ::.  //\n *
:::..           ..::::::             :===-::::::..             .:::.
//\n *      .:::..              .:::             -=-:::.
.::::.    //\n *         .::::.           .:::                 ..
.::::.       //\n *           .::::.        .:::
..:::.           //\n *               .:::.       .:::
.::::.              //\n *                 .:::..  .:::
..:::..                 //\n *                 .::::.:::
.::::.                     //\n *                  ..::::
..:::..                        //\n *                          .:
.::::.                            //\n *
:::::.::::.                               //\n *
::::::::.                                  //\n *
:::::::.                                   //\n *
.::::.                                    //\n *
```

```
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/            //\n * ----------------
----------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity
^0.8.17;\n\ninterface IOperatorFilterRegistry {\n    function
isOperatorAllowed(address registrant, address operator) external view
returns (bool);\n    function register(address registrant) external;\n
function registerAndSubscribe(address registrant, address subscription)
external;\n    function registerAndCopyEntries(address registrant,
address registrantToCopy) external;\n    function updateOperator(address
registrant, address operator, bool filtered) external;\n    function
updateOperators(address registrant, address[] calldata operators, bool
filtered) external;\n    function updateCodeHash(address registrant,
bytes32 codehash, bool filtered) external;\n    function
updateCodeHashes(address registrant, bytes32[] calldata codeHashes, bool
filtered) external;\n    function subscribe(address registrant, address
registrantToSubscribe) external;\n    function unsubscribe(address
registrant, bool copyExistingEntries) external;\n    function
subscriptionOf(address addr) external returns (address registrant);\n
function subscribers(address registrant) external returns (address[]
memory);\n    function subscriberAt(address registrant, uint256 index)
external returns (address);\n    function copyEntriesOf(address
registrant, address registrantToCopy) external;\n    function
isOperatorFiltered(address registrant, address operator) external returns
(bool);\n    function isCodeHashOfFiltered(address registrant, address
operatorWithCode) external returns (bool);\n    function
isCodeHashFiltered(address registrant, bytes32 codeHash) external returns
(bool);\n    function filteredOperators(address addr) external returns
(address[] memory);\n    function filteredCodeHashes(address addr)
external returns (bytes32[] memory);\n    function
filteredOperatorAt(address registrant, uint256 index) external returns
(address);\n    function filteredCodeHashAt(address registrant, uint256
index) external returns (bytes32);\n    function isRegistered(address
addr) external view returns (bool);\n    function codeHashOf(address
addr) external view returns (bytes32);\n}"
    },
    "@nfts2me/contracts/ownable/NFTOwnableUpgradeable.sol": {
      "content": "/** -----------------------------------------------------
------------------------- //\n *
//\n *                                        .:::.
//\n *                                     .:::::::.
//\n *                                     ::::::::.
//\n *                                  .:::::::::.
//\n *                              ..:::.              ..
//\n *                           .::::.                 ::::..
//\n *                       ..:::..                    ::::::::.
```

```
//\n *                    .::::.                        :::.  ..:::.
//\n *                ..:::..                           :::.      .:::.
//\n *             .::::.                               :::.
.:::..           //\n *          .:::..               ..
:::.            .::::.        //\n *      .::::.               ..:::=-
::::               ..:::.     //\n *     :::.               .:::::::===:
::::::.               .::::   //\n *    .::.            .:::::::::::=====.
::::::::::.             .::.  //\n *    .::
.:::::::::::::::=======.          :::::::::::::..          ::.  //\n *
.::        .::::::::::::::::========-         :::::::::::::::::
::.  //\n *    .::        .::::::::::::::::==========:
:::::::::::::::::        ::.  //\n *    .::
.::::::::::::::::============:     :::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   :::::::::::::::::
::.  //\n *    .::        .::::::::::::::::===============-.
:::::::::::::::::        ::.  //\n *    .::
.::::::::::::::::=================::::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==================-::::::::::::::::
::.  //\n *    .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *    .::
.::::::::::::::::==================-::::::::::::::::        ::.  //\n *
.::        .:::::::::::::::::=================-::::::::::::::::
::.  //\n *    .::        .:::::::::::::::: .================-
::::::::::::::::        ::.  //\n *    .::          .::::::::::::::::
.==============-::::::::::::::::        ::.  //\n *    .::
.::::::::::::::::     :============-::::::::::::::::        ::.  //\n *
.::         .::::::::::::::::       :==========-::::::::::::::::
::.  //\n *    .::        .::::::::::::::::        .-========-
::::::::::::::::        ::.  //\n *    .::          .::::::::::::::
.=======-::::::::::::::.         ::.  //\n *    .::.
.::::::::::            .=====-::::::::::..            ::.  //\n *
:::..              ..::::::              :===-::::::..              .:::.
//\n *       .:::..               .:::                -=-:::.
.::::.    //\n *          .::::.           .:::                 ..
.::::.       //\n *             .::::.         .:::
..:::.           //\n *                 .:::.      .:::
.::::.              //\n *                    .:::..  .:::
..:::..                 //\n *                       .::::.:::
.::::.                     //\n *                          ..::::
..:::..                        //\n *                              .:
.::::.                            //\n *
:::::.::::.                               //\n *
::::::::.                                  //\n *
:::::::.                                   //\n *
.::::.                                    //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
```

https://nfts2me.com/terms-of-service/   //\n * ----------------
-- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.17;\n\nimport
\"@openzeppelin/contracts-
upgradeable/proxy/utils/Initializable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/utils/ContextUpgradeable.sol\";\n\n/// @title NFTs2Me.com
Smart Contracts\n/// @author The NFTs2Me Team\n/// @notice Read our terms
of service\n/// @custom:security-contact security@nfts2me.com\n///
@custom:terms-of-service https://nfts2me.com/terms-of-service/\n///
@custom:website https://nfts2me.com/\nabstract contract
NFTOwnableUpgradeable is Initializable, ContextUpgradeable {\n   /**\n
* @dev Throws if called by any account other than the owner.\n   */\n
modifier onlyOwner() {\n   _checkOwner();\n   _;\n   }\n\n
modifier onlyStrictOwner() {\n   _checkStrictOwner();\n   _;\n
}\n\n   modifier onlyOwnerOrN2M() {\n   _checkOwnerOrN2M();\n
_;\n   }\n\n   modifier onlyN2M() {\n   _checkN2M();\n   _;\n
}\n\n   /// @notice Returns the address of the current collection
owner.\n   /// @return The address of the owner.\n   function owner()
public view virtual returns (address);\n   function _strictOwner()
internal view virtual returns (address);\n   function
_getN2MFeeAddress() internal view virtual returns (address);\n\n
function _checkOwner() internal view virtual {\n   require(owner()
== msg.sender, \"Ownable: caller is not the owner\");\n   }\n\n
function _checkStrictOwner() internal view virtual {\n
require(_strictOwner() == msg.sender, \"Ownable: caller is not the
owner\");\n   }\n\n   function _checkOwnerOrN2M() internal view virtual
{\n   require(owner() == msg.sender || _getN2MFeeAddress() ==
msg.sender, \"Ownable: caller is not the owner\");\n   }\n\n   function
_checkN2M() internal view virtual {\n   require(_getN2MFeeAddress()
== msg.sender, \"Ownable: caller is not the owner\");\n   }\n\n}"
    },
    "@openzeppelin/contracts-
upgradeable/governance/utils/IVotesUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.5.0) (governance/utils/IVotes.sol)\npragma
solidity ^0.8.0;\n\n/**\n * @dev Common interface for {ERC20Votes},
{ERC721Votes}, and other {Votes}-enabled contracts.\n *\n * _Available
since v4.5._\n */\ninterface IVotesUpgradeable {\n   /**\n   * @dev
Emitted when an account changes their delegate.\n   */\n   event
DelegateChanged(address indexed delegator, address indexed fromDelegate,
address indexed toDelegate);\n\n   /**\n   * @dev Emitted when a token
transfer or delegate change results in changes to a delegate's number of
votes.\n   */\n   event DelegateVotesChanged(address indexed delegate,
uint256 previousBalance, uint256 newBalance);\n\n   /**\n   * @dev
Returns the current amount of votes that `account` has.\n   */\n
function getVotes(address account) external view returns (uint256);\n\n
/**\n   * @dev Returns the amount of votes that `account` had at the
end of a past block (`blockNumber`).\n   */\n   function
getPastVotes(address account, uint256 blockNumber) external view returns
(uint256);\n\n   /**\n   * @dev Returns the total supply of votes
available at the end of a past block (`blockNumber`).\n   *\n   *
NOTE: This value is the sum of all available votes, which is not
necessarily the sum of all delegated votes.\n   * Votes that have not

been delegated are still part of total supply, even though they would not participate in a\n   * vote.\n   */\n   function getPastTotalSupply(uint256 blockNumber) external view returns (uint256);\n\n   /**\n   * @dev Returns the delegate that `account` has chosen.\n   */\n   function delegates(address account) external view returns (address);\n\n   /**\n   * @dev Delegates votes from the sender to `delegatee`.\n   */\n   function delegate(address delegatee) external;\n\n   /**\n   * @dev Delegates votes from signer to `delegatee`.\n   */\n   function delegateBySig(\n   address delegatee,\n   uint256 nonce,\n   uint256 expiry,\n   uint8 v,\n   bytes32 r,\n   bytes32 s\n   ) external;\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/governance/utils/VotesUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.8.0) (governance/utils/Votes.sol)\npragma solidity ^0.8.0;\n\nimport \"../../utils/ContextUpgradeable.sol\";\nimport \"../../utils/CountersUpgradeable.sol\";\nimport \"../../utils/CheckpointsUpgradeable.sol\";\nimport \"../../utils/cryptography/EIP712Upgradeable.sol\";\nimport \"./IVotesUpgradeable.sol\";\nimport \"../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev This is a base abstract contract that tracks voting units, which are a measure of voting power that can be\n * transferred, and provides a system of vote delegation, where an account can delegate its voting units to a sort of\n * \"representative\" that will pool delegated voting units from different accounts and can then use it to vote in\n * decisions. In fact, voting units _must_ be delegated in order to count as actual votes, and an account has to\n * delegate those votes to itself if it wishes to participate in decisions and does not have a trusted representative.\n *\n * This contract is often combined with a token contract such that voting units correspond to token units. For an\n * example, see {ERC721Votes}.\n *\n * The full history of delegate votes is tracked on-chain so that governance protocols can consider votes as distributed\n * at a particular block number to protect against flash loans and double voting. The opt-in delegate system makes the\n * cost of this history tracking optional.\n *\n * When using this module the derived contract must implement {_getVotingUnits} (for example, make it return\n * {ERC721-balanceOf}), and can use {_transferVotingUnits} to track a change in the distribution of those units (in the\n * previous example, it would be included in {ERC721-_beforeTokenTransfer}).\n *\n * _Available since v4.5._\n */\nabstract contract VotesUpgradeable is Initializable, IVotesUpgradeable, ContextUpgradeable, EIP712Upgradeable {\n   function __Votes_init() internal onlyInitializing {\n   }\n\n   function __Votes_init_unchained() internal onlyInitializing {\n   }\n   using CheckpointsUpgradeable for CheckpointsUpgradeable.History;\n   using CountersUpgradeable for CountersUpgradeable.Counter;\n\n   bytes32 private constant _DELEGATION_TYPEHASH =\n   keccak256(\"Delegation(address delegatee,uint256 nonce,uint256 expiry)\");\n\n   mapping(address => address) private _delegation;\n   mapping(address => CheckpointsUpgradeable.History) private _delegateCheckpoints;\n   CheckpointsUpgradeable.History private _totalCheckpoints;\n\n   mapping(address => CountersUpgradeable.Counter)

```
private _nonces;\n\n    /**\n     * @dev Returns the current amount of
votes that `account` has.\n     */\n    function getVotes(address
account) public view virtual override returns (uint256) {\n        return
_delegateCheckpoints[account].latest();\n    }\n\n    /**\n     * @dev
Returns the amount of votes that `account` had at the end of a past block
(`blockNumber`).\n     *\n     * Requirements:\n     *\n     * -
`blockNumber` must have been already mined\n     */\n    function
getPastVotes(address account, uint256 blockNumber) public view virtual
override returns (uint256) {\n        return
_delegateCheckpoints[account].getAtProbablyRecentBlock(blockNumber);\n
}\n\n    /**\n     * @dev Returns the total supply of votes available at
the end of a past block (`blockNumber`).\n     *\n     * NOTE: This value
is the sum of all available votes, which is not necessarily the sum of
all delegated votes.\n     * Votes that have not been delegated are still
part of total supply, even though they would not participate in a\n     *
vote.\n     *\n     * Requirements:\n     *\n     * - `blockNumber` must
have been already mined\n     */\n    function getPastTotalSupply(uint256
blockNumber) public view virtual override returns (uint256) {\n
require(blockNumber < block.number, \"Votes: block not yet mined\");\n
return _totalCheckpoints.getAtProbablyRecentBlock(blockNumber);\n
}\n\n    /**\n     * @dev Returns the current total supply of votes.\n
*/\n    function _getTotalSupply() internal view virtual returns
(uint256) {\n        return _totalCheckpoints.latest();\n    }\n\n
/**\n     * @dev Returns the delegate that `account` has chosen.\n
*/\n    function delegates(address account) public view virtual override
returns (address) {\n        return _delegation[account];\n    }\n\n
/**\n     * @dev Delegates votes from the sender to `delegatee`.\n
*/\n    function delegate(address delegatee) public virtual override {\n
address account = _msgSender();\n        _delegate(account, delegatee);\n
}\n\n    /**\n     * @dev Delegates votes from signer to `delegatee`.\n
*/\n    function delegateBySig(\n        address delegatee,\n
uint256 nonce,\n        uint256 expiry,\n        uint8 v,\n
bytes32 r,\n        bytes32 s\n    ) public virtual override {\n
require(block.timestamp <= expiry, \"Votes: signature expired\");\n
address signer = ECDSAUpgradeable.recover(\n
_hashTypedDataV4(keccak256(abi.encode(_DELEGATION_TYPEHASH, delegatee,
nonce, expiry))),\n            v,\n            r,\n            s\n
);\n        require(nonce == _useNonce(signer), \"Votes: invalid
nonce\");\n        _delegate(signer, delegatee);\n    }\n\n    /**\n
* @dev Delegate all of `account`'s voting units to `delegatee`.\n     *\n
* Emits events {IVotes-DelegateChanged} and {IVotes-
DelegateVotesChanged}.\n     */\n    function _delegate(address account,
address delegatee) internal virtual {\n        address oldDelegate =
delegates(account);\n        _delegation[account] = delegatee;\n\n
emit DelegateChanged(account, oldDelegate, delegatee);\n
_moveDelegateVotes(oldDelegate, delegatee, _getVotingUnits(account));\n
}\n\n    /**\n     * @dev Transfers, mints, or burns voting units. To
register a mint, `from` should be zero. To register a burn, `to`\n     *
should be zero. Total supply of voting units will be adjusted with mints
and burns.\n     */\n    function _transferVotingUnits(\n        address
from,\n        address to,\n        uint256 amount\n    ) internal
virtual {\n        if (from == address(0)) {\n
_totalCheckpoints.push(_add, amount);\n        }\n        if (to ==
address(0)) {\n            _totalCheckpoints.push(_subtract, amount);\n
```

}\n   _moveDelegateVotes(delegates(from), delegates(to), amount);\n   }\n\n   /**\n   * @dev Moves delegated votes from one delegate to another.\n   */\n   function _moveDelegateVotes(\n   address from,\n   address to,\n   uint256 amount\n   ) private {\n   if (from != to && amount > 0) {\n   if (from != address(0)) {\n   (uint256 oldValue, uint256 newValue) = _delegateCheckpoints[from].push(_subtract, amount);\n   emit DelegateVotesChanged(from, oldValue, newValue);\n   }\n   if (to != address(0)) {\n   (uint256 oldValue, uint256 newValue) = _delegateCheckpoints[to].push(_add, amount);\n   emit DelegateVotesChanged(to, oldValue, newValue);\n   }\n   }\n   }\n\n   function _add(uint256 a, uint256 b) private pure returns (uint256) {\n   return a + b;\n   }\n\n   function _subtract(uint256 a, uint256 b) private pure returns (uint256) {\n   return a - b;\n   }\n\n   /**\n   * @dev Consumes a nonce.\n   *\n   * Returns the current value and increments nonce.\n   */\n   function _useNonce(address owner) internal virtual returns (uint256 current) {\n   CountersUpgradeable.Counter storage nonce = _nonces[owner];\n   current = nonce.current();\n   nonce.increment();\n   }\n\n   /**\n   * @dev Returns an address nonce.\n   */\n   function nonces(address owner) public view virtual returns (uint256) {\n   return _nonces[owner].current();\n   }\n\n   /**\n   * @dev Returns the contract's {EIP712} domain separator.\n   */\n   // solhint-disable-next-line func-name-mixedcase\n   function DOMAIN_SEPARATOR() external view returns (bytes32) {\n   return _domainSeparatorV4();\n   }\n\n   /**\n   * @dev Must return the voting units held by an account.\n   */\n   function _getVotingUnits(address) internal view virtual returns (uint256);\n\n   /**\n   * @dev This empty reserved space is put in place to allow future versions to add new\n   * variables without shifting down storage in the inheritance chain.\n   * See https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n   */\n   uint256[46] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/interfaces/IERC2981Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.6.0) (interfaces/IERC2981.sol)\n\npragma solidity ^0.8.0;\n\nimport \"../utils/introspection/IERC165Upgradeable.sol\";\n\n/**\n * @dev Interface for the NFT Royalty Standard.\n *\n * A standardized way to retrieve royalty payment information for non-fungible tokens (NFTs) to enable universal\n * support for royalty payments across all NFT marketplaces and ecosystem participants.\n *\n * _Available since v4.5._\n */\ninterface IERC2981Upgradeable is IERC165Upgradeable {\n   /**\n   * @dev Returns how much royalty is owed and to whom, based on a sale price that may be denominated in any unit of\n   * exchange. The royalty amount is denominated and should be paid in that same unit of exchange.\n   */\n   function royaltyInfo(uint256 tokenId, uint256 salePrice)\n   external\n   view\n   returns (address receiver, uint256 royaltyAmount);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol": {

```
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.1) (proxy/utils/Initializable.sol)\n\npragma
solidity ^0.8.2;\n\nimport
\"../../utils/AddressUpgradeable.sol\";\n\n/**\n * @dev This is a base
contract to aid in writing upgradeable contracts, or any kind of contract
that will be deployed\n * behind a proxy. Since proxied contracts do not
make use of a constructor, it's common to move constructor logic to an\n
* external initializer function, usually called `initialize`. It then
becomes necessary to protect this initializer\n * function so it can only
be called once. The {initializer} modifier provided by this contract will
have this effect.\n *\n * The initialization functions use a version
number. Once a version number is used, it is consumed and cannot be\n *
reused. This mechanism prevents re-execution of each \"step\" but allows
the creation of new initialization steps in\n * case an upgrade adds a
module that needs to be initialized.\n *\n * For example:\n *\n * [.hljs-
theme-light.nopadding]\n * ```\n * contract MyToken is ERC20Upgradeable
{\n *     function initialize() initializer public {\n *
__ERC20_init(\"MyToken\", \"MTK\");\n *     }\n * }\n * contract
MyTokenV2 is MyToken, ERC20PermitUpgradeable {\n *     function
initializeV2() reinitializer(2) public {\n *
__ERC20Permit_init(\"MyToken\");\n *     }\n * }\n * ```\n *\n * TIP: To
avoid leaving the proxy in an uninitialized state, the initializer
function should be called as early as\n * possible by providing the
encoded function call as the `_data` argument to {ERC1967Proxy-
constructor}.\n *\n * CAUTION: When used with inheritance, manual care
must be taken to not invoke a parent initializer twice, or to ensure\n *
that all initializers are idempotent. This is not verified automatically
as constructors are by Solidity.\n *\n * [CAUTION]\n * ====\n * Avoid
leaving a contract uninitialized.\n *\n * An uninitialized contract can
be taken over by an attacker. This applies to both a proxy and its
implementation\n * contract, which may impact the proxy. To prevent the
implementation contract from being used, you should invoke\n * the
{_disableInitializers} function in the constructor to automatically lock
it when it is deployed:\n *\n * [.hljs-theme-light.nopadding]\n * ```\n *
/// @custom:oz-upgrades-unsafe-allow constructor\n * constructor() {\n *
_disableInitializers();\n * }\n * ```\n * ====\n */\nabstract contract
Initializable {\n    /**\n     * @dev Indicates that the contract has
been initialized.\n     * @custom:oz-retyped-from bool\n     */\n
uint8 private _initialized;\n\n    /**\n     * @dev Indicates that the
contract is in the process of being initialized.\n     */\n    bool
private _initializing;\n\n    /**\n     * @dev Triggered when the
contract has been initialized or reinitialized.\n     */\n    event
Initialized(uint8 version);\n\n    /**\n     * @dev A modifier that
defines a protected initializer function that can be invoked at most
once. In its scope,\n     * `onlyInitializing` functions can be used to
initialize parent contracts.\n     *\n     * Similar to
`reinitializer(1)`, except that functions marked with `initializer` can
be nested in the context of a\n     * constructor.\n     *\n     * Emits
an {Initialized} event.\n     */\n    modifier initializer() {\n
bool isTopLevelCall = !_initializing;\n        require(\n
(isTopLevelCall && _initialized < 1) ||
(!AddressUpgradeable.isContract(address(this)) && _initialized == 1),\n
\"Initializable: contract is already initialized\"\n        );\n
_initialized = 1;\n        if (isTopLevelCall) {\n
```

_initializing = true;\n   }\n   _;\n   if (isTopLevelCall) {\n   _initializing = false;\n   emit Initialized(1);\n   }\n   }\n\n   /**\n   * @dev A modifier that defines a protected reinitializer function that can be invoked at most once, and only if the\n   * contract hasn't been initialized to a greater version before. In its scope, `onlyInitializing` functions can be\n   * used to initialize parent contracts.\n   *\n   * A reinitializer may be used after the original initialization step. This is essential to configure modules that\n   * are added through upgrades and that require initialization.\n   *\n   * When `version` is 1, this modifier is similar to `initializer`, except that functions marked with `reinitializer`\n   * cannot be nested. If one is invoked in the context of another, execution will revert.\n   *\n   * Note that versions can jump in increments greater than 1; this implies that if multiple reinitializers coexist in\n   * a contract, executing them in the right order is up to the developer or operator.\n   *\n   * WARNING: setting the version to 255 will prevent any future reinitialization.\n   *\n   * Emits an {Initialized} event.\n   */\n   modifier reinitializer(uint8 version) {\n   require(!_initializing && _initialized < version, \"Initializable: contract is already initialized\");\n   _initialized = version;\n   _initializing = true;\n   _;\n   _initializing = false;\n   emit Initialized(version);\n   }\n\n   /**\n   * @dev Modifier to protect an initialization function so that it can only be invoked by functions with the\n   * {initializer} and {reinitializer} modifiers, directly or indirectly.\n   */\n   modifier onlyInitializing() {\n   require(_initializing, \"Initializable: contract is not initializing\");\n   _;\n   }\n\n   /**\n   * @dev Locks the contract, preventing any future reinitialization. This cannot be part of an initializer call.\n   * Calling this in the constructor of a contract will prevent that contract from being initialized or reinitialized\n   * to any version. It is recommended to use this to lock implementation contracts that are designed to be called\n   * through proxies.\n   *\n   * Emits an {Initialized} event the first time it is successfully executed.\n   */\n   function _disableInitializers() internal virtual {\n   require(!_initializing, \"Initializable: contract is initializing\");\n   if (_initialized < type(uint8).max) {\n   _initialized = type(uint8).max;\n   emit Initialized(type(uint8).max);\n   }\n   }\n\n   /**\n   * @dev Returns the highest version that has been initialized. See {reinitializer}.\n   */\n   function _getInitializedVersion() internal view returns (uint8) {\n   return _initialized;\n   }\n\n   /**\n   * @dev Returns `true` if the contract is currently initializing. See {onlyInitializing}.\n   */\n   function _isInitializing() internal view returns (bool) {\n   return _initializing;\n   }\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC20/IERC20.sol)\n\npragma solidity ^0.8.0;\n\n/**\n * @dev Interface of the ERC20 standard as defined in the EIP.\n */\ninterface IERC20Upgradeable {\n   /**\n   * @dev Emitted when `value` tokens are moved from one account (`from`) to\n

```
* another (`to`).\n     *\n     * Note that `value` may be zero.\n
*/\n    event Transfer(address indexed from, address indexed to, uint256
value);\n\n    /**\n     * @dev Emitted when the allowance of a `spender`
for an `owner` is set by\n     * a call to {approve}. `value` is the new
allowance.\n     */\n    event Approval(address indexed owner, address
indexed spender, uint256 value);\n\n    /**\n     * @dev Returns the
amount of tokens in existence.\n     */\n    function totalSupply()
external view returns (uint256);\n\n    /**\n     * @dev Returns the
amount of tokens owned by `account`.\n     */\n    function
balanceOf(address account) external view returns (uint256);\n\n    /**\n
* @dev Moves `amount` tokens from the caller's account to `to`.\n     *\n
* Returns a boolean value indicating whether the operation succeeded.\n
*\n     * Emits a {Transfer} event.\n     */\n    function
transfer(address to, uint256 amount) external returns (bool);\n\n
/**\n     * @dev Returns the remaining number of tokens that `spender`
will be\n     * allowed to spend on behalf of `owner` through
{transferFrom}. This is\n     * zero by default.\n     *\n     * This
value changes when {approve} or {transferFrom} are called.\n     */\n
function allowance(address owner, address spender) external view returns
(uint256);\n\n    /**\n     * @dev Sets `amount` as the allowance of
`spender` over the caller's tokens.\n     *\n     * Returns a boolean
value indicating whether the operation succeeded.\n     *\n     *
IMPORTANT: Beware that changing an allowance with this method brings the
risk\n     * that someone may use both the old and the new allowance by
unfortunate\n     * transaction ordering. One possible solution to
mitigate this race\n     * condition is to first reduce the spender's
allowance to 0 and set the\n     * desired value afterwards:\n     *
https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729\n
*\n     * Emits an {Approval} event.\n     */\n    function
approve(address spender, uint256 amount) external returns (bool);\n\n
/**\n     * @dev Moves `amount` tokens from `from` to `to` using the\n
* allowance mechanism. `amount` is then deducted from the caller's\n
* allowance.\n     *\n     * Returns a boolean value indicating whether
the operation succeeded.\n     *\n     * Emits a {Transfer} event.\n
*/\n    function transferFrom(\n        address from,\n        address
to,\n        uint256 amount\n    ) external returns (bool);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-
IERC20PermitUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts v4.4.1 (token/ERC20/extensions/draft-
IERC20Permit.sol)\n\npragma solidity ^0.8.0;\n\n/**\n * @dev Interface of
the ERC20 Permit extension allowing approvals to be made via signatures,
as defined in\n * https://eips.ethereum.org/EIPS/eip-2612[EIP-2612].\n
*\n * Adds the {permit} method, which can be used to change an account's
ERC20 allowance (see {IERC20-allowance}) by\n * presenting a message
signed by the account. By not relying on {IERC20-approve}, the token
holder account doesn't\n * need to send a transaction, and thus is not
required to hold Ether at all.\n */\ninterface IERC20PermitUpgradeable
{\n    /**\n     * @dev Sets `value` as the allowance of `spender` over
``owner``'s tokens,\n     * given ``owner``'s signed approval.\n     *\n
* IMPORTANT: The same issues {IERC20-approve} has related to
transaction\n     * ordering also apply here.\n     *\n     * Emits an
{Approval} event.\n     *\n     * Requirements:\n     *\n     * -
```

`spender` cannot be the zero address.\n   * - `deadline` must be a timestamp in the future.\n   * - `v`, `r` and `s` must be a valid `secp256k1` signature from `owner`\n   * over the EIP712-formatted function arguments.\n   * - the signature must use ``owner``'s current nonce (see {nonces}).\n   *\n   * For more information on the signature format, see the\n   * https://eips.ethereum.org/EIPS/eip-2612#specification[relevant EIP\n   * section].\n   */\n   function permit(\n   address owner,\n   address spender,\n   uint256 value,\n   uint256 deadline,\n   uint8 v,\n   bytes32 r,\n   bytes32 s\n   ) external;\n\n   /**\n   * @dev Returns the current nonce for `owner`. This value must be\n   * included whenever a signature is generated for {permit}.\n   *\n   * Every successful call to {permit} increases ``owner``'s nonce by one. This\n   * prevents a signature from being used multiple times.\n   */\n   function nonces(address owner) external view returns (uint256);\n\n   /**\n   * @dev Returns the domain separator used in the encoding of the signature for {permit}, as defined by {EIP712}.\n   */\n   // solhint-disable-next-line func-name-mixedcase\n   function DOMAIN_SEPARATOR() external view returns (bytes32);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.8.0) (token/ERC20/utils/SafeERC20.sol)\n\npragma solidity ^0.8.0;\n\nimport \"../IERC20Upgradeable.sol\";\nimport \"../extensions/draft-IERC20PermitUpgradeable.sol\";\nimport \"../../../utils/AddressUpgradeable.sol\";\n\n/**\n * @title SafeERC20\n * @dev Wrappers around ERC20 operations that throw on failure (when the token\n * contract returns false). Tokens that return no value (and instead revert or\n * throw on failure) are also supported, non-reverting calls are assumed to be\n * successful.\n * To use this library you can add a `using SafeERC20 for IERC20;` statement to your contract,\n * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.\n */\nlibrary SafeERC20Upgradeable {\n   using AddressUpgradeable for address;\n\n   function safeTransfer(\n   IERC20Upgradeable token,\n   address to,\n   uint256 value\n   ) internal {\n   _callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to, value));\n   }\n\n   function safeTransferFrom(\n   IERC20Upgradeable token,\n   address from,\n   address to,\n   uint256 value\n   ) internal {\n   _callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to, value));\n   }\n\n   /**\n   * @dev Deprecated. This function has issues similar to the ones found in\n   * {IERC20-approve}, and its usage is discouraged.\n   *\n   * Whenever possible, use {safeIncreaseAllowance} and\n   * {safeDecreaseAllowance} instead.\n   */\n   function safeApprove(\n   IERC20Upgradeable token,\n   address spender,\n   uint256 value\n   ) internal {\n   // safeApprove should only be called when setting an initial allowance,\n   // or when resetting it to zero. To increase and decrease it, use\n   // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'\n   require(\n   (value == 0) || (token.allowance(address(this), spender) == 0),\n   \"SafeERC20: approve from non-zero to non-

zero allowance\"\n   );\n   _callOptionalReturn(token,
abi.encodeWithSelector(token.approve.selector, spender, value));\n
}\n\n   function safeIncreaseAllowance(\n   IERC20Upgradeable
token,\n   address spender,\n   uint256 value\n   ) internal
{\n   uint256 newAllowance = token.allowance(address(this), spender)
+ value;\n   _callOptionalReturn(token,
abi.encodeWithSelector(token.approve.selector, spender, newAllowance));\n
}\n\n   function safeDecreaseAllowance(\n   IERC20Upgradeable
token,\n   address spender,\n   uint256 value\n   ) internal
{\n   unchecked {\n   uint256 oldAllowance =
token.allowance(address(this), spender);\n
require(oldAllowance >= value, \"SafeERC20: decreased allowance below
zero\");\n   uint256 newAllowance = oldAllowance - value;\n
_callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector,
spender, newAllowance));\n   }\n   }\n\n   function safePermit(\n
IERC20PermitUpgradeable token,\n   address owner,\n   address
spender,\n   uint256 value,\n   uint256 deadline,\n
uint8 v,\n   bytes32 r,\n   bytes32 s\n   ) internal {\n
uint256 nonceBefore = token.nonces(owner);\n   token.permit(owner,
spender, value, deadline, v, r, s);\n   uint256 nonceAfter =
token.nonces(owner);\n   require(nonceAfter == nonceBefore + 1,
\"SafeERC20: permit did not succeed\");\n   }\n\n   /**\n   * @dev
Imitates a Solidity high-level call (i.e. a regular function call to a
contract), relaxing the requirement\n   * on the return value: the
return value is optional (but if data is returned, it must not be
false).\n   * @param token The token targeted by the call.\n   *
@param data The call data (encoded using abi.encode or one of its
variants).\n   */\n   function _callOptionalReturn(IERC20Upgradeable
token, bytes memory data) private {\n   // We need to perform a low
level call here, to bypass Solidity's return data size checking
mechanism, since\n   // we're implementing it ourselves. We use
{Address-functionCall} to perform this call, which verifies that\n
// the target address contains contract code and also asserts for success
in the low-level call.\n\n   bytes memory returndata =
address(token).functionCall(data, \"SafeERC20: low-level call
failed\");\n   if (returndata.length > 0) {\n   // Return
data is optional\n   require(abi.decode(returndata, (bool)),
\"SafeERC20: ERC20 operation did not succeed\");\n   }\n   }\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.2) (token/ERC721/ERC721.sol)\n\npragma
solidity ^0.8.0;\n\nimport \"./IERC721Upgradeable.sol\";\nimport
\"./IERC721ReceiverUpgradeable.sol\";\nimport
\"./extensions/IERC721MetadataUpgradeable.sol\";\nimport
\"../../utils/AddressUpgradeable.sol\";\nimport
\"../../utils/ContextUpgradeable.sol\";\nimport
\"../../utils/StringsUpgradeable.sol\";\nimport
\"../../utils/introspection/ERC165Upgradeable.sol\";\nimport
\"../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev Implementation
of https://eips.ethereum.org/EIPS/eip-721[ERC721] Non-Fungible Token
Standard, including\n * the Metadata extension, but not including the
Enumerable extension, which is available separately as\n *

```
{ERC721Enumerable}.\n */\ncontract ERC721Upgradeable is Initializable,
ContextUpgradeable, ERC165Upgradeable, IERC721Upgradeable,
IERC721MetadataUpgradeable {\n    using AddressUpgradeable for address;\n
using StringsUpgradeable for uint256;\n\n    // Token name\n    string
private _name;\n\n    // Token symbol\n    string private _symbol;\n\n
// Mapping from token ID to owner address\n    mapping(uint256 =>
address) private _owners;\n\n    // Mapping owner address to token
count\n    mapping(address => uint256) private _balances;\n\n    //
Mapping from token ID to approved address\n    mapping(uint256 =>
address) private _tokenApprovals;\n\n    // Mapping from owner to
operator approvals\n    mapping(address => mapping(address => bool))
private _operatorApprovals;\n\n    /**\n     * @dev Initializes the
contract by setting a `name` and a `symbol` to the token collection.\n
*/\n    function __ERC721_init(string memory name_, string memory
symbol_) internal onlyInitializing {\n
__ERC721_init_unchained(name_, symbol_);\n    }\n\n    function
__ERC721_init_unchained(string memory name_, string memory symbol_)
internal onlyInitializing {\n        _name = name_;\n        _symbol =
symbol_;\n    }\n\n    /**\n     * @dev See {IERC165-
supportsInterface}.\n     */\n    function supportsInterface(bytes4
interfaceId) public view virtual override(ERC165Upgradeable,
IERC165Upgradeable) returns (bool) {\n        return\n
interfaceId == type(IERC721Upgradeable).interfaceId ||\n
interfaceId == type(IERC721MetadataUpgradeable).interfaceId ||\n
super.supportsInterface(interfaceId);\n    }\n\n    /**\n     * @dev See
{IERC721-balanceOf}.\n     */\n    function balanceOf(address owner)
public view virtual override returns (uint256) {\n        require(owner
!= address(0), \"ERC721: address zero is not a valid owner\");\n
return _balances[owner];\n    }\n\n    /**\n     * @dev See {IERC721-
ownerOf}.\n     */\n    function ownerOf(uint256 tokenId) public view
virtual override returns (address) {\n        address owner =
_ownerOf(tokenId);\n        require(owner != address(0), \"ERC721:
invalid token ID\");\n        return owner;\n    }\n\n    /**\n     *
@dev See {IERC721Metadata-name}.\n     */\n    function name() public
view virtual override returns (string memory) {\n        return _name;\n
}\n\n    /**\n     * @dev See {IERC721Metadata-symbol}.\n     */\n
function symbol() public view virtual override returns (string memory)
{\n        return _symbol;\n    }\n\n    /**\n     * @dev See
{IERC721Metadata-tokenURI}.\n     */\n    function tokenURI(uint256
tokenId) public view virtual override returns (string memory) {\n
_requireMinted(tokenId);\n\n        string memory baseURI = _baseURI();\n
return bytes(baseURI).length > 0 ? string(abi.encodePacked(baseURI,
tokenId.toString())) : \"\";\n    }\n\n    /**\n     * @dev Base URI for
computing {tokenURI}. If set, the resulting URI for each\n     * token
will be the concatenation of the `baseURI` and the `tokenId`. Empty\n
* by default, can be overridden in child contracts.\n     */\n
function _baseURI() internal view virtual returns (string memory) {\n
return \"\";\n    }\n\n    /**\n     * @dev See {IERC721-approve}.\n
*/\n    function approve(address to, uint256 tokenId) public virtual
override {\n        address owner = ERC721Upgradeable.ownerOf(tokenId);\n
require(to != owner, \"ERC721: approval to current owner\");\n\n
require(\n            _msgSender() == owner || isApprovedForAll(owner,
_msgSender()),\n            \"ERC721: approve caller is not token owner
or approved for all\"\n        );\n\n        _approve(to, tokenId);\n
}\n\n    /**\n     * @dev See {IERC721-getApproved}.\n     */\n
function getApproved(uint256 tokenId) public view virtual override
returns (address) {\n        _requireMinted(tokenId);\n\n        return
_tokenApprovals[tokenId];\n    }\n\n    /**\n     * @dev See {IERC721-
setApprovalForAll}.\n     */\n    function setApprovalForAll(address
operator, bool approved) public virtual override {\n
_setApprovalForAll(_msgSender(), operator, approved);\n    }\n\n    /**\n
* @dev See {IERC721-isApprovedForAll}.\n     */\n    function
isApprovedForAll(address owner, address operator) public view virtual
override returns (bool) {\n        return
_operatorApprovals[owner][operator];\n    }\n\n    /**\n     * @dev See
{IERC721-transferFrom}.\n     */\n    function transferFrom(\n
address from,\n        address to,\n        uint256 tokenId\n    ) public
virtual override {\n        //solhint-disable-next-line max-line-length\n
require(_isApprovedOrOwner(_msgSender(), tokenId), \"ERC721: caller is
not token owner or approved\");\n\n        _transfer(from, to,
tokenId);\n    }\n\n    /**\n     * @dev See {IERC721-
safeTransferFrom}.\n     */\n    function safeTransferFrom(\n
address from,\n        address to,\n        uint256 tokenId\n    ) public
virtual override {\n        safeTransferFrom(from, to, tokenId, \"\");\n
}\n\n    /**\n     * @dev See {IERC721-safeTransferFrom}.\n     */\n
function safeTransferFrom(\n        address from,\n        address to,\n
uint256 tokenId,\n        bytes memory data\n    ) public virtual
override {\n        require(_isApprovedOrOwner(_msgSender(), tokenId),
\"ERC721: caller is not token owner or approved\");\n
_safeTransfer(from, to, tokenId, data);\n    }\n\n    /**\n     * @dev
Safely transfers `tokenId` token from `from` to `to`, checking first that
contract recipients\n     * are aware of the ERC721 protocol to prevent
tokens from being forever locked.\n     *\n     * `data` is additional
data, it has no specified format and it is sent in call to `to`.\n
*\n     * This internal function is equivalent to {safeTransferFrom}, and
can be used to e.g.\n     * implement alternative mechanisms to perform
token transfer, such as signature-based.\n     *\n     * Requirements:\n
*\n     * - `from` cannot be the zero address.\n     * - `to` cannot be
the zero address.\n     * - `tokenId` token must exist and be owned by
`from`.\n     * - If `to` refers to a smart contract, it must implement
{IERC721Receiver-onERC721Received}, which is called upon a safe
transfer.\n     *\n     * Emits a {Transfer} event.\n     */\n
function _safeTransfer(\n        address from,\n        address to,\n
uint256 tokenId,\n        bytes memory data\n    ) internal virtual {\n
_transfer(from, to, tokenId);\n
require(_checkOnERC721Received(from, to, tokenId, data), \"ERC721:
transfer to non ERC721Receiver implementer\");\n    }\n\n    /**\n     *
@dev Returns the owner of the `tokenId`. Does NOT revert if token doesn't
exist\n     */\n    function _ownerOf(uint256 tokenId) internal view
virtual returns (address) {\n        return _owners[tokenId];\n    }\n\n
/**\n     * @dev Returns whether `tokenId` exists.\n     *\n     * Tokens
can be managed by their owner or approved accounts via {approve} or
{setApprovalForAll}.\n     *\n     * Tokens start existing when they are
minted (`_mint`),\n     * and stop existing when they are burned
(`_burn`).\n     */\n    function _exists(uint256 tokenId) internal view
virtual returns (bool) {\n        return _ownerOf(tokenId) !=
address(0);\n    }\n\n    /**\n     * @dev Returns whether `spender` is
allowed to manage `tokenId`.\n     *\n     * Requirements:\n     *\n
```

```
* - `tokenId` must exist.\n     */\n    function
_isApprovedOrOwner(address spender, uint256 tokenId) internal view
virtual returns (bool) {\n        address owner =
ERC721Upgradeable.ownerOf(tokenId);\n        return (spender == owner ||
isApprovedForAll(owner, spender) || getApproved(tokenId) == spender);\n
}\n\n    /**\n     * @dev Safely mints `tokenId` and transfers it to
`to`.\n     *\n     * Requirements:\n     *\n     * - `tokenId` must not
exist.\n     * - If `to` refers to a smart contract, it must implement
{IERC721Receiver-onERC721Received}, which is called upon a safe
transfer.\n     *\n     * Emits a {Transfer} event.\n     */\n
function _safeMint(address to, uint256 tokenId) internal virtual {\n
_safeMint(to, tokenId, \"\");\n    }\n\n    /**\n     * @dev Same as
{xref-ERC721-_safeMint-address-uint256-}[`_safeMint`], with an additional
`data` parameter which is\n     * forwarded in {IERC721Receiver-
onERC721Received} to contract recipients.\n     */\n    function
_safeMint(\n        address to,\n        uint256 tokenId,\n        bytes
memory data\n    ) internal virtual {\n        _mint(to, tokenId);\n
require(\n            _checkOnERC721Received(address(0), to, tokenId,
data),\n            \"ERC721: transfer to non ERC721Receiver
implementer\"\n        );\n    }\n\n    /**\n     * @dev Mints `tokenId`
and transfers it to `to`.\n     *\n     * WARNING: Usage of this method
is discouraged, use {_safeMint} whenever possible\n     *\n     *
Requirements:\n     *\n     * - `tokenId` must not exist.\n     * - `to`
cannot be the zero address.\n     *\n     * Emits a {Transfer} event.\n
*/\n    function _mint(address to, uint256 tokenId) internal virtual {\n
require(to != address(0), \"ERC721: mint to the zero address\");\n
require(!_exists(tokenId), \"ERC721: token already minted\");\n\n
_beforeTokenTransfer(address(0), to, tokenId, 1);\n\n        // Check
that tokenId was not minted by `_beforeTokenTransfer` hook\n
require(!_exists(tokenId), \"ERC721: token already minted\");\n\n
unchecked {\n            // Will not overflow unless all 2**256 token ids
are minted to the same owner.\n            // Given that tokens are
minted one by one, it is impossible in practice that\n            // this
ever happens. Might change if we allow batch minting.\n            // The
ERC fails to describe this case.\n            _balances[to] += 1;\n
}\n\n        _owners[tokenId] = to;\n\n        emit Transfer(address(0),
to, tokenId);\n\n        _afterTokenTransfer(address(0), to, tokenId,
1);\n    }\n\n    /**\n     * @dev Destroys `tokenId`.\n     * The
approval is cleared when the token is burned.\n     * This is an internal
function that does not check if the sender is authorized to operate on
the token.\n     *\n     * Requirements:\n     *\n     * - `tokenId` must
exist.\n     *\n     * Emits a {Transfer} event.\n     */\n    function
_burn(uint256 tokenId) internal virtual {\n        address owner =
ERC721Upgradeable.ownerOf(tokenId);\n\n
_beforeTokenTransfer(owner, address(0), tokenId, 1);\n\n        // Update
ownership in case tokenId was transferred by `_beforeTokenTransfer`
hook\n        owner = ERC721Upgradeable.ownerOf(tokenId);\n\n        //
Clear approvals\n        delete _tokenApprovals[tokenId];\n\n
unchecked {\n            // Cannot overflow, as that would require more
tokens to be burned/transferred\n            // out than the owner
initially received through minting and transferring in.\n
_balances[owner] -= 1;\n        }\n        delete _owners[tokenId];\n\n
emit Transfer(owner, address(0), tokenId);\n\n
_afterTokenTransfer(owner, address(0), tokenId, 1);\n    }\n\n    /**\n
```

```
* @dev Transfers `tokenId` from `from` to `to`.\n     *  As opposed to
{transferFrom}, this imposes no restrictions on msg.sender.\n     *\n
* Requirements:\n     *\n     * - `to` cannot be the zero address.\n
* - `tokenId` token must be owned by `from`.\n     *\n     * Emits a
{Transfer} event.\n     */\n    function _transfer(\n        address
from,\n        address to,\n        uint256 tokenId\n    ) internal
virtual {\n        require(ERC721Upgradeable.ownerOf(tokenId) == from,
\"ERC721: transfer from incorrect owner\");\n        require(to !=
address(0), \"ERC721: transfer to the zero address\");\n\n
_beforeTokenTransfer(from, to, tokenId, 1);\n\n        // Check that
tokenId was not transferred by `_beforeTokenTransfer` hook\n
require(ERC721Upgradeable.ownerOf(tokenId) == from, \"ERC721: transfer
from incorrect owner\");\n\n        // Clear approvals from the previous
owner\n        delete _tokenApprovals[tokenId];\n\n        unchecked {\n
// `_balances[from]` cannot overflow for the same reason as described in
`_burn`:\n            // `from`'s balance is the number of token held,
which is at least one before the current\n            // transfer.\n
// `_balances[to]` could overflow in the conditions described in `_mint`.
That would require\n            // all 2**256 token ids to be minted,
which in practice is impossible.\n            _balances[from] -= 1;\n
_balances[to] += 1;\n        }\n        _owners[tokenId] = to;\n\n
emit Transfer(from, to, tokenId);\n\n        _afterTokenTransfer(from,
to, tokenId, 1);\n    }\n\n    /**\n     * @dev Approve `to` to operate
on `tokenId`\n     *\n     * Emits an {Approval} event.\n     */\n
function _approve(address to, uint256 tokenId) internal virtual {\n
_tokenApprovals[tokenId] = to;\n        emit
Approval(ERC721Upgradeable.ownerOf(tokenId), to, tokenId);\n    }\n\n
/**\n     * @dev Approve `operator` to operate on all of `owner` tokens\n
*\n     * Emits an {ApprovalForAll} event.\n     */\n    function
_setApprovalForAll(\n        address owner,\n        address operator,\n
bool approved\n    ) internal virtual {\n        require(owner !=
operator, \"ERC721: approve to caller\");\n
_operatorApprovals[owner][operator] = approved;\n        emit
ApprovalForAll(owner, operator, approved);\n    }\n\n    /**\n     * @dev
Reverts if the `tokenId` has not been minted yet.\n     */\n    function
_requireMinted(uint256 tokenId) internal view virtual {\n
require(_exists(tokenId), \"ERC721: invalid token ID\");\n    }\n\n
/**\n     * @dev Internal function to invoke {IERC721Receiver-
onERC721Received} on a target address.\n     * The call is not executed
if the target address is not a contract.\n     *\n     * @param from
address representing the previous owner of the given token ID\n     *
@param to target address that will receive the tokens\n     * @param
tokenId uint256 ID of the token to be transferred\n     * @param data
bytes optional data to send along with the call\n     * @return bool
whether the call correctly returned the expected magic value\n     */\n
function _checkOnERC721Received(\n        address from,\n        address
to,\n        uint256 tokenId,\n        bytes memory data\n    ) private
returns (bool) {\n        if (to.isContract()) {\n            try
IERC721ReceiverUpgradeable(to).onERC721Received(_msgSender(), from,
tokenId, data) returns (bytes4 retval) {\n                return retval
== IERC721ReceiverUpgradeable.onERC721Received.selector;\n            }
catch (bytes memory reason) {\n                if (reason.length == 0)
{\n                    revert(\"ERC721: transfer to non ERC721Receiver
implementer\");\n                } else {\n                    ///
@solidity memory-safe-assembly\n                    assembly {\n
revert(add(32, reason), mload(reason))\n                    }\n
}\n            }\n        } else {\n            return true;\n        }\n
}\n\n    /**\n     * @dev Hook that is called before any token transfer.
This includes minting and burning. If {ERC721Consecutive} is\n     *
used, the hook may be called as part of a consecutive (batch) mint, as
indicated by `batchSize` greater than 1.\n     *\n     * Calling
conditions:\n     *\n     * - When `from` and `to` are both non-zero,
``from``'s tokens will be transferred to `to`.\n     * - When `from` is
zero, the tokens will be minted for `to`.\n     * - When `to` is zero,
``from``'s tokens will be burned.\n     * - `from` and `to` are never
both zero.\n     * - `batchSize` is non-zero.\n     *\n     * To learn
more about hooks, head to xref:ROOT:extending-contracts.adoc#using-
hooks[Using Hooks].\n     */\n    function _beforeTokenTransfer(\n
address from,\n        address to,\n        uint256 firstTokenId,\n
uint256 batchSize\n    ) internal virtual {}\n\n    /**\n     * @dev Hook
that is called after any token transfer. This includes minting and
burning. If {ERC721Consecutive} is\n     * used, the hook may be called
as part of a consecutive (batch) mint, as indicated by `batchSize`
greater than 1.\n     *\n     * Calling conditions:\n     *\n     * -
When `from` and `to` are both non-zero, ``from``'s tokens were
transferred to `to`.\n     * - When `from` is zero, the tokens were
minted for `to`.\n     * - When `to` is zero, ``from``'s tokens were
burned.\n     * - `from` and `to` are never both zero.\n     * -
`batchSize` is non-zero.\n     *\n     * To learn more about hooks, head
to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].\n
*/\n    function _afterTokenTransfer(\n        address from,\n
address to,\n        uint256 firstTokenId,\n        uint256 batchSize\n
) internal virtual {}\n\n    /**\n     * @dev Unsafe write access to the
balances, used by extensions that \"mint\" tokens using an {ownerOf}
override.\n     *\n     * WARNING: Anyone calling this MUST ensure that
the balances remain consistent with the ownership. The invariant\n     *
being that for any address `a` the value returned by `balanceOf(a)` must
be equal to the number of tokens such\n     * that `ownerOf(tokenId)` is
`a`.\n     */\n    // solhint-disable-next-line func-name-mixedcase\n
function __unsafe_increaseBalance(address account, uint256 amount)
internal {\n        _balances[account] += amount;\n    }\n\n    /**\n
* @dev This empty reserved space is put in place to allow future versions
to add new\n     * variables without shifting down storage in the
inheritance chain.\n     * See
https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n
*/\n    uint256[44] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.6.0)
(token/ERC721/IERC721Receiver.sol)\n\npragma solidity ^0.8.0;\n\n/**\n *
@title ERC721 token receiver interface\n * @dev Interface for any
contract that wants to support safeTransfers\n * from ERC721 asset
contracts.\n */\ninterface IERC721ReceiverUpgradeable {\n    /**\n     *
@dev Whenever an {IERC721} `tokenId` token is transferred to this
contract via {IERC721-safeTransferFrom}\n     * by `operator` from
`from`, this function is called.\n     *\n     * It must return its
```

Solidity selector to confirm the token transfer.\n   * If any other value is returned or the interface is not implemented by the recipient, the transfer will be reverted.\n   *\n   * The selector can be obtained in Solidity with `IERC721Receiver.onERC721Received.selector`.\n   */\n   function onERC721Received(\n   address operator,\n   address from,\n   uint256 tokenId,\n   bytes calldata data\n   ) external returns (bytes4);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.8.0) (token/ERC721/IERC721.sol)\n\npragma solidity ^0.8.0;\n\nimport \"../../utils/introspection/IERC165Upgradeable.sol\";\n\n/**\n * @dev Required interface of an ERC721 compliant contract.\n */\ninterface IERC721Upgradeable is IERC165Upgradeable {\n   /**\n   * @dev Emitted when `tokenId` token is transferred from `from` to `to`.\n   */\n   event Transfer(address indexed from, address indexed to, uint256 indexed tokenId);\n\n   /**\n   * @dev Emitted when `owner` enables `approved` to manage the `tokenId` token.\n   */\n   event Approval(address indexed owner, address indexed approved, uint256 indexed tokenId);\n\n   /**\n   * @dev Emitted when `owner` enables or disables (`approved`) `operator` to manage all of its assets.\n   */\n   event ApprovalForAll(address indexed owner, address indexed operator, bool approved);\n\n   /**\n   * @dev Returns the number of tokens in ``owner``'s account.\n   */\n   function balanceOf(address owner) external view returns (uint256 balance);\n\n   /**\n   * @dev Returns the owner of the `tokenId` token.\n   *\n   * Requirements:\n   *\n   * - `tokenId` must exist.\n   */\n   function ownerOf(uint256 tokenId) external view returns (address owner);\n\n   /**\n   * @dev Safely transfers `tokenId` token from `from` to `to`.\n   *\n   * Requirements:\n   *\n   * - `from` cannot be the zero address.\n   * - `to` cannot be the zero address.\n   * - `tokenId` token must exist and be owned by `from`.\n   * - If the caller is not `from`, it must be approved to move this token by either {approve} or {setApprovalForAll}.\n   * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.\n   *\n   * Emits a {Transfer} event.\n   */\n   function safeTransferFrom(\n   address from,\n   address to,\n   uint256 tokenId,\n   bytes calldata data\n   ) external;\n\n   /**\n   * @dev Safely transfers `tokenId` token from `from` to `to`, checking first that contract recipients\n   * are aware of the ERC721 protocol to prevent tokens from being forever locked.\n   *\n   * Requirements:\n   *\n   * - `from` cannot be the zero address.\n   * - `to` cannot be the zero address.\n   * - `tokenId` token must exist and be owned by `from`.\n   * - If the caller is not `from`, it must have been allowed to move this token by either {approve} or {setApprovalForAll}.\n   * - If `to` refers to a smart contract, it must implement {IERC721Receiver-onERC721Received}, which is called upon a safe transfer.\n   *\n   * Emits a {Transfer} event.\n   */\n   function safeTransferFrom(\n   address from,\n   address to,\n   uint256 tokenId\n   ) external;\n\n   /**\n   * @dev Transfers `tokenId` token from `from` to `to`.\n   *\n   * WARNING: Note that the caller is responsible to confirm that the recipient is capable of

receiving ERC721\n   * or else they may be permanently lost. Usage of {safeTransferFrom} prevents loss, though the caller must\n   * understand this adds an external call which potentially creates a reentrancy vulnerability.\n   *\n   * Requirements:\n   *\n   * - `from` cannot be the zero address.\n   * - `to` cannot be the zero address.\n   * - `tokenId` token must be owned by `from`.\n   * - If the caller is not `from`, it must be approved to move this token by either {approve} or {setApprovalForAll}.\n   *\n   * Emits a {Transfer} event.\n   */\n   function transferFrom(\n   address from,\n   address to,\n   uint256 tokenId\n   ) external;\n\n   /**\n   * @dev Gives permission to `to` to transfer `tokenId` token to another account.\n   * The approval is cleared when the token is transferred.\n   *\n   * Only a single account can be approved at a time, so approving the zero address clears previous approvals.\n   *\n   * Requirements:\n   *\n   * - The caller must own the token or be an approved operator.\n   * - `tokenId` must exist.\n   *\n   * Emits an {Approval} event.\n   */\n   function approve(address to, uint256 tokenId) external;\n\n   /**\n   * @dev Approve or remove `operator` as an operator for the caller.\n   * Operators can call {transferFrom} or {safeTransferFrom} for any token owned by the caller.\n   *\n   * Requirements:\n   *\n   * - The `operator` cannot be the caller.\n   *\n   * Emits an {ApprovalForAll} event.\n   */\n   function setApprovalForAll(address operator, bool _approved) external;\n\n   /**\n   * @dev Returns the account approved for `tokenId` token.\n   *\n   * Requirements:\n   *\n   * - `tokenId` must exist.\n   */\n   function getApproved(uint256 tokenId) external view returns (address operator);\n\n   /**\n   * @dev Returns if the `operator` is allowed to manage all of the assets of `owner`.\n   *\n   * See {setApprovalForAll}\n   */\n   function isApprovedForAll(address owner, address operator) external view returns (bool);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.7.0) (token/ERC721/extensions/ERC721URIStorage.sol)\n\npragma solidity ^0.8.0;\n\nimport \"../ERC721Upgradeable.sol\";\nimport \"../../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev ERC721 token with storage based token URI management.\n */\nabstract contract ERC721URIStorageUpgradeable is Initializable, ERC721Upgradeable {\n   function __ERC721URIStorage_init() internal onlyInitializing {\n   }\n\n   function __ERC721URIStorage_init_unchained() internal onlyInitializing {\n   }\n   using StringsUpgradeable for uint256;\n\n   // Optional mapping for token URIs\n   mapping(uint256 => string) private _tokenURIs;\n\n   /**\n   * @dev See {IERC721Metadata-tokenURI}.\n   */\n   function tokenURI(uint256 tokenId) public view virtual override returns (string memory) {\n   _requireMinted(tokenId);\n\n   string memory _tokenURI = _tokenURIs[tokenId];\n   string memory base = _baseURI();\n\n   // If there is no base URI, return the token URI.\n   if (bytes(base).length == 0) {\n   return _tokenURI;\n   }\n   // If both are set, concatenate the baseURI and tokenURI (via abi.encodePacked).\n   if (bytes(_tokenURI).length > 0) {\n   return string(abi.encodePacked(base, _tokenURI));\n   }\n\n   return

```
super.tokenURI(tokenId);\n    }\n\n    /**\n     * @dev Sets `_tokenURI`
as the tokenURI of `tokenId`.\n     *\n     * Requirements:\n     *\n
* - `tokenId` must exist.\n     */\n    function _setTokenURI(uint256
tokenId, string memory _tokenURI) internal virtual {\n
require(_exists(tokenId), \"ERC721URIStorage: URI set of nonexistent
token\");\n        _tokenURIs[tokenId] = _tokenURI;\n    }\n\n    /**\n
* @dev See {ERC721-_burn}. This override additionally checks to see if
a\n     * token-specific URI was set for the token, and if so, it deletes
the token URI from\n     * the storage mapping.\n     */\n    function
_burn(uint256 tokenId) internal virtual override {\n
super._burn(tokenId);\n\n        if (bytes(_tokenURIs[tokenId]).length !=
0) {\n            delete _tokenURIs[tokenId];\n        }\n    }\n\n
/**\n     * @dev This empty reserved space is put in place to allow
future versions to add new\n     * variables without shifting down
storage in the inheritance chain.\n     * See
https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n
*/\n    uint256[49] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721VotesUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0)
(token/ERC721/extensions/ERC721Votes.sol)\n\npragma solidity
^0.8.0;\n\nimport \"../ERC721Upgradeable.sol\";\nimport
\"../../../governance/utils/VotesUpgradeable.sol\";\nimport
\"../../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev Extension of
ERC721 to support voting and delegation as implemented by {Votes}, where
each individual NFT counts\n * as 1 vote unit.\n *\n * Tokens do not
count as votes until they are delegated, because votes must be tracked
which incurs an additional cost\n * on every transfer. Token holders can
either delegate to a trusted representative who will decide how to make
use of\n * the votes in governance decisions, or they can delegate to
themselves to be their own representative.\n *\n * _Available since
v4.5._\n */\nabstract contract ERC721VotesUpgradeable is Initializable,
ERC721Upgradeable, VotesUpgradeable {\n    function __ERC721Votes_init()
internal onlyInitializing {\n    }\n\n    function
__ERC721Votes_init_unchained() internal onlyInitializing {\n    }\n
/**\n     * @dev See {ERC721-_afterTokenTransfer}. Adjusts votes when
tokens are transferred.\n     *\n     * Emits a {IVotes-
DelegateVotesChanged} event.\n     */\n    function
_afterTokenTransfer(\n        address from,\n        address to,\n
uint256 firstTokenId,\n        uint256 batchSize\n    ) internal virtual
override {\n        _transferVotingUnits(from, to, batchSize);\n
super._afterTokenTransfer(from, to, firstTokenId, batchSize);\n    }\n\n
/**\n     * @dev Returns the balance of `account`.\n     */\n    function
_getVotingUnits(address account) internal view virtual override returns
(uint256) {\n        return balanceOf(account);\n    }\n\n    /**\n     *
@dev This empty reserved space is put in place to allow future versions
to add new\n     * variables without shifting down storage in the
inheritance chain.\n     * See
https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n
*/\n    uint256[50] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts v4.4.1 (token/ERC721/extensions/IERC721Metadata.sol)\n\npragma
solidity ^0.8.0;\n\nimport \"../IERC721Upgradeable.sol\";\n\n/**\n *
@title ERC-721 Non-Fungible Token Standard, optional metadata extension\n
* @dev See https://eips.ethereum.org/EIPS/eip-721\n */\ninterface
IERC721MetadataUpgradeable is IERC721Upgradeable {\n    /**\n     * @dev
Returns the token collection name.\n     */\n    function name() external
view returns (string memory);\n\n    /**\n     * @dev Returns the token
collection symbol.\n     */\n    function symbol() external view returns
(string memory);\n\n    /**\n     * @dev Returns the Uniform Resource
Identifier (URI) for `tokenId` token.\n     */\n    function
tokenURI(uint256 tokenId) external view returns (string memory);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0) (utils/Address.sol)\n\npragma solidity
^0.8.1;\n\n/**\n * @dev Collection of functions related to the address
type\n */\nlibrary AddressUpgradeable {\n    /**\n     * @dev Returns
true if `account` is a contract.\n     *\n     * [IMPORTANT]\n     *
====\n     * It is unsafe to assume that an address for which this
function returns\n     * false is an externally-owned account (EOA) and
not a contract.\n     *\n     * Among others, `isContract` will return
false for the following\n     * types of addresses:\n     *\n     *  - an
externally-owned account\n     *  - a contract in construction\n     *  -
an address where a contract will be created\n     *  - an address where a
contract lived, but was destroyed\n     * ====\n     *\n     *
[IMPORTANT]\n     * ====\n     * You shouldn't rely on `isContract` to
protect against flash loan attacks!\n     *\n     * Preventing calls from
contracts is highly discouraged. It breaks composability, breaks support
for smart wallets\n     * like Gnosis Safe, and does not provide security
since it can be circumvented by calling from a contract\n     *
constructor.\n     * ====\n     */\n    function isContract(address
account) internal view returns (bool) {\n        // This method relies on
extcodesize/address.code.length, which returns 0\n        // for
contracts in construction, since the code is only stored at the end\n
// of the constructor execution.\n\n        return account.code.length >
0;\n    }\n\n    /**\n     * @dev Replacement for Solidity's `transfer`:
sends `amount` wei to\n     * `recipient`, forwarding all available gas
and reverting on errors.\n     *\n     *
https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost\n
* of certain opcodes, possibly making contracts go over the 2300 gas
limit\n     * imposed by `transfer`, making them unable to receive funds
via\n     * `transfer`. {sendValue} removes this limitation.\n     *\n
* https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-
transfer-now/[Learn more].\n     *\n     * IMPORTANT: because control is
transferred to `recipient`, care must be\n     * taken to not create
reentrancy vulnerabilities. Consider using\n     * {ReentrancyGuard} or
the\n     * ¹https://solidity.readthedocs.io/en/v0.5.11/security-
considerations.html#use-the-checks-effects-interactions-pattern[checks-
effects-interactions pattern].\n     */\n    function sendValue(address
payable recipient, uint256 amount) internal {\n
require(address(this).balance >= amount, \"Address: insufficient
balance\");\n\n        (bool success, ) = recipient.call{value:
amount}(\"\");\n        require(success, \"Address: unable to send value,
recipient may have reverted\");\n    }\n\n    /**\n     * @dev Performs a
Solidity function call using a low level `call`. A\n     * plain `call`
is an unsafe replacement for a function call: use this\n     * function
instead.\n     *\n     * If `target` reverts with a revert reason, it is
bubbled up by this\n     * function (like regular Solidity function
calls).\n     *\n     * Returns the raw returned data. To convert to the
expected return value,\n     * use
https://solidity.readthedocs.io/en/latest/units-and-global-
variables.html?highlight=abi.decode#abi-encoding-and-decoding-
functions[`abi.decode`].\n     *\n     * Requirements:\n     *\n     * -
`target` must be a contract.\n     * - calling `target` with `data` must
not revert.\n     *\n     * _Available since v3.1._\n     */\n
function functionCall(address target, bytes memory data) internal returns
(bytes memory) {\n        return functionCallWithValue(target, data, 0,
\"Address: low-level call failed\");\n    }\n\n    /**\n     * @dev Same
as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with\n
* `errorMessage` as a fallback revert reason when `target` reverts.\n
*\n     * _Available since v3.1._\n     */\n    function functionCall(\n
address target,\n        bytes memory data,\n        string memory
errorMessage\n    ) internal returns (bytes memory) {\n        return
functionCallWithValue(target, data, 0, errorMessage);\n    }\n\n    /**\n
* @dev Same as {xref-Address-functionCall-address-bytes-
}[`functionCall`],\n     * but also transferring `value` wei to
`target`.\n     *\n     * Requirements:\n     *\n     * - the calling
contract must have an ETH balance of at least `value`.\n     * - the
called Solidity function must be `payable`.\n     *\n     * _Available
since v3.1._\n     */\n    function functionCallWithValue(\n
address target,\n        bytes memory data,\n        uint256 value\n    )
internal returns (bytes memory) {\n        return
functionCallWithValue(target, data, value, \"Address: low-level call with
value failed\");\n    }\n\n    /**\n     * @dev Same as {xref-Address-
functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`],
but\n     * with `errorMessage` as a fallback revert reason when `target`
reverts.\n     *\n     * _Available since v3.1._\n     */\n    function
functionCallWithValue(\n        address target,\n        bytes memory
data,\n        uint256 value,\n        string memory errorMessage\n    )
internal returns (bytes memory) {\n        require(address(this).balance
>= value, \"Address: insufficient balance for call\");\n        (bool
success, bytes memory returndata) = target.call{value: value}(data);\n
return verifyCallResultFromTarget(target, success, returndata,
errorMessage);\n    }\n\n    /**\n     * @dev Same as {xref-Address-
functionCall-address-bytes-}[`functionCall`],\n     * but performing a
static call.\n     *\n     * _Available since v3.3._\n     */\n
function functionStaticCall(address target, bytes memory data) internal
view returns (bytes memory) {\n        return functionStaticCall(target,
data, \"Address: low-level static call failed\");\n    }\n\n    /**\n
* @dev Same as {xref-Address-functionCall-address-bytes-string-
}[`functionCall`],\n     * but performing a static call.\n     *\n     *
_Available since v3.3._\n     */\n    function functionStaticCall(\n
address target,\n        bytes memory data,\n        string memory
errorMessage\n    ) internal view returns (bytes memory) {\n        (bool
success, bytes memory returndata) = target.staticcall(data);\n
```

```
return verifyCallResultFromTarget(target, success, returndata,
errorMessage);\n    }\n\n    /**\n     * @dev Tool to verify that a low
level call to smart-contract was successful, and revert (either by
bubbling\n     * the revert reason or using the provided one) in case of
unsuccessful call or if target was not a contract.\n     *\n     *
_Available since v4.8._\n     */\n    function
verifyCallResultFromTarget(\n        address target,\n        bool
success,\n        bytes memory returndata,\n        string memory
errorMessage\n    ) internal view returns (bytes memory) {\n        if
(success) {\n            if (returndata.length == 0) {\n
// only check isContract if the call was successful and the return data
is empty\n                // otherwise we already know that it was a
contract\n                require(isContract(target), \"Address: call to
non-contract\");\n            }\n            return returndata;\n
} else {\n            _revert(returndata, errorMessage);\n        }\n
}\n\n    /**\n     * @dev Tool to verify that a low level call was
successful, and revert if it wasn't, either by bubbling the\n     *
revert reason or using the provided one.\n     *\n     * _Available since
v4.3._\n     */\n    function verifyCallResult(\n        bool success,\n
bytes memory returndata,\n        string memory errorMessage\n    )
internal pure returns (bytes memory) {\n        if (success) {\n
return returndata;\n        } else {\n            _revert(returndata,
errorMessage);\n        }\n    }\n\n    function _revert(bytes memory
returndata, string memory errorMessage) private pure {\n        // Look
for revert reason and bubble it up if present\n        if
(returndata.length > 0) {\n            // The easiest way to bubble the
revert reason is using memory via assembly\n            /// @solidity
memory-safe-assembly\n            assembly {\n                let
returndata_size := mload(returndata)\n                revert(add(32,
returndata), returndata_size)\n            }\n        } else {\n
revert(errorMessage);\n        }\n    }\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/CheckpointsUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.1) (utils/Checkpoints.sol)\n// This file was
procedurally generated from
scripts/generate/templates/Checkpoints.js.\n\npragma solidity
^0.8.0;\n\nimport \"./math/MathUpgradeable.sol\";\nimport
\"./math/SafeCastUpgradeable.sol\";\n\n/**\n * @dev This library defines
the `History` struct, for checkpointing values as they change at
different points in\n * time, and later looking up past values by block
number. See {Votes} as an example.\n *\n * To create a history of
checkpoints define a variable type `Checkpoints.History` in your
contract, and store a new\n * checkpoint for the current transaction
block using the {push} function.\n *\n * _Available since v4.5._\n
*/\nlibrary CheckpointsUpgradeable {\n    struct History {\n
Checkpoint[] _checkpoints;\n    }\n\n    struct Checkpoint {\n
uint32 _blockNumber;\n        uint224 _value;\n    }\n\n    /**\n     *
@dev Returns the value at a given block number. If a checkpoint is not
available at that block, the closest one\n     * before it is returned,
or zero otherwise. Because the number returned corresponds to that at the
end of the\n     * block, the requested block number must be in the past,
excluding the current block.\n     */\n    function getAtBlock(History
storage self, uint256 blockNumber) internal view returns (uint256) {\n
require(blockNumber < block.number, \"Checkpoints: block not yet
mined\");\n        uint32 key =
SafeCastUpgradeable.toUint32(blockNumber);\n\n        uint256 len =
self._checkpoints.length;\n        uint256 pos =
_upperBinaryLookup(self._checkpoints, key, 0, len);\n        return pos
== 0 ? 0 : _unsafeAccess(self._checkpoints, pos - 1)._value;\n    }\n\n
/**\n     * @dev Returns the value at a given block number. If a
checkpoint is not available at that block, the closest one\n     * before
it is returned, or zero otherwise. Similar to {upperLookup} but optimized
for the case when the searched\n     * checkpoint is probably \"recent\",
defined as being among the last sqrt(N) checkpoints where N is the number
of\n     * checkpoints.\n     */\n    function
getAtProbablyRecentBlock(History storage self, uint256 blockNumber)
internal view returns (uint256) {\n        require(blockNumber <
block.number, \"Checkpoints: block not yet mined\");\n        uint32 key
= SafeCastUpgradeable.toUint32(blockNumber);\n\n        uint256 len =
self._checkpoints.length;\n\n        uint256 low = 0;\n        uint256
high = len;\n\n        if (len > 5) {\n            uint256 mid = len -
MathUpgradeable.sqrt(len);\n            if (key <
_unsafeAccess(self._checkpoints, mid)._blockNumber) {\n
high = mid;\n            } else {\n                low = mid + 1;\n
}\n        }\n\n        uint256 pos =
_upperBinaryLookup(self._checkpoints, key, low, high);\n\n        return
pos == 0 ? 0 : _unsafeAccess(self._checkpoints, pos - 1)._value;\n
}\n\n    /**\n     * @dev Pushes a value onto a History so that it is
stored as the checkpoint for the current block.\n     *\n     * Returns
previous value and new value.\n     */\n    function push(History storage
self, uint256 value) internal returns (uint256, uint256) {\n
return _insert(self._checkpoints,
SafeCastUpgradeable.toUint32(block.number),
SafeCastUpgradeable.toUint224(value));\n    }\n\n    /**\n     * @dev
Pushes a value onto a History, by updating the latest value using binary
operation `op`. The new value will\n     * be set to `op(latest,
delta)`.\n     *\n     * Returns previous value and new value.\n     */\n
function push(\n        History storage self,\n        function(uint256,
uint256) view returns (uint256) op,\n        uint256 delta\n    )
internal returns (uint256, uint256) {\n        return push(self,
op(latest(self), delta));\n    }\n\n    /**\n     * @dev Returns the
value in the most recent checkpoint, or zero if there are no
checkpoints.\n     */\n    function latest(History storage self) internal
view returns (uint224) {\n        uint256 pos =
self._checkpoints.length;\n        return pos == 0 ? 0 :
_unsafeAccess(self._checkpoints, pos - 1)._value;\n    }\n\n    /**\n
* @dev Returns whether there is a checkpoint in the structure (i.e. it is
not empty), and if so the key and value\n     * in the most recent
checkpoint.\n     */\n    function latestCheckpoint(History storage
self)\n        internal\n        view\n        returns (\n
bool exists,\n            uint32 _blockNumber,\n            uint224
_value\n        )\n    {\n        uint256 pos =
self._checkpoints.length;\n        if (pos == 0) {\n            return
(false, 0, 0);\n        } else {\n            Checkpoint memory ckpt =
_unsafeAccess(self._checkpoints, pos - 1);\n            return (true,
ckpt._blockNumber, ckpt._value);\n        }\n    }\n\n    /**\n     *
```

```
@dev Returns the number of checkpoint.\n     */\n    function
length(History storage self) internal view returns (uint256) {\n
return self._checkpoints.length;\n    }\n\n    /**\n     * @dev Pushes a
(`key`, `value`) pair into an ordered list of checkpoints, either by
inserting a new checkpoint,\n     * or by updating the last one.\n
*/\n    function _insert(\n        Checkpoint[] storage self,\n
uint32 key,\n        uint224 value\n    ) private returns (uint224,
uint224) {\n        uint256 pos = self.length;\n\n        if (pos > 0)
{\n            // Copying to memory is important here.\n
Checkpoint memory last = _unsafeAccess(self, pos - 1);\n\n            //
Checkpoints keys must be increasing.\n
require(last._blockNumber <= key, \"Checkpoint: invalid key\");\n\n
// Update or push new checkpoint\n            if (last._blockNumber ==
key) {\n                _unsafeAccess(self, pos - 1)._value = value;\n
} else {\n                self.push(Checkpoint({_blockNumber: key,
_value: value}));\n            }\n            return (last._value,
value);\n        } else {\n
self.push(Checkpoint({_blockNumber: key, _value: value}));\n
return (0, value);\n        }\n    }\n\n    /**\n     * @dev Return the
index of the oldest checkpoint whose key is greater than the search key,
or `high` if there is none.\n     * `low` and `high` define a section
where to do the search, with inclusive `low` and exclusive `high`.\n
*\n     * WARNING: `high` should not be greater than the array's
length.\n     */\n    function _upperBinaryLookup(\n        Checkpoint[]
storage self,\n        uint32 key,\n        uint256 low,\n        uint256
high\n    ) private view returns (uint256) {\n        while (low < high)
{\n            uint256 mid = MathUpgradeable.average(low, high);\n
if (_unsafeAccess(self, mid)._blockNumber > key) {\n                high
= mid;\n            } else {\n                low = mid + 1;\n
}\n        }\n        return high;\n    }\n\n    /**\n     * @dev Return
the index of the oldest checkpoint whose key is greater or equal than the
search key, or `high` if there is none.\n     * `low` and `high` define a
section where to do the search, with inclusive `low` and exclusive
`high`.\n     *\n     * WARNING: `high` should not be greater than the
array's length.\n     */\n    function _lowerBinaryLookup(\n
Checkpoint[] storage self,\n        uint32 key,\n        uint256 low,\n
uint256 high\n    ) private view returns (uint256) {\n        while (low
< high) {\n            uint256 mid = MathUpgradeable.average(low,
high);\n            if (_unsafeAccess(self, mid)._blockNumber < key) {\n
low = mid + 1;\n            } else {\n                high = mid;\n
}\n        }\n        return high;\n    }\n\n    /**\n     * @dev Access
an element of the array without performing bounds check. The position is
assumed to be within bounds.\n     */\n    function
_unsafeAccess(Checkpoint[] storage self, uint256 pos) private pure
returns (Checkpoint storage result) {\n        assembly {\n
mstore(0, self.slot)\n            result.slot := add(keccak256(0, 0x20),
pos)\n        }\n    }\n\n    struct Trace224 {\n        Checkpoint224[]
_checkpoints;\n    }\n\n    struct Checkpoint224 {\n        uint32
_key;\n        uint224 _value;\n    }\n\n    /**\n     * @dev Pushes a
(`key`, `value`) pair into a Trace224 so that it is stored as the
checkpoint.\n     *\n     * Returns previous value and new value.\n
*/\n    function push(\n        Trace224 storage self,\n        uint32
key,\n        uint224 value\n    ) internal returns (uint224, uint224)
{\n        return _insert(self._checkpoints, key, value);\n    }\n\n
```

```
/**\n     * @dev Returns the value in the oldest checkpoint with key
greater or equal than the search key, or zero if there is none.\n
*/\n    function lowerLookup(Trace224 storage self, uint32 key) internal
view returns (uint224) {\n        uint256 len =
self._checkpoints.length;\n        uint256 pos =
_lowerBinaryLookup(self._checkpoints, key, 0, len);\n        return pos
== len ? 0 : _unsafeAccess(self._checkpoints, pos)._value;\n    }\n\n
/**\n     * @dev Returns the value in the most recent checkpoint with key
lower or equal than the search key.\n     */\n    function
upperLookup(Trace224 storage self, uint32 key) internal view returns
(uint224) {\n        uint256 len = self._checkpoints.length;\n
uint256 pos = _upperBinaryLookup(self._checkpoints, key, 0, len);\n
return pos == 0 ? 0 : _unsafeAccess(self._checkpoints, pos - 1)._value;\n
}\n\n    /**\n     * @dev Returns the value in the most recent
checkpoint, or zero if there are no checkpoints.\n     */\n    function
latest(Trace224 storage self) internal view returns (uint224) {\n
uint256 pos = self._checkpoints.length;\n        return pos == 0 ? 0 :
_unsafeAccess(self._checkpoints, pos - 1)._value;\n    }\n\n    /**\n
* @dev Returns whether there is a checkpoint in the structure (i.e. it is
not empty), and if so the key and value\n     * in the most recent
checkpoint.\n     */\n    function latestCheckpoint(Trace224 storage
self)\n        internal\n        view\n        returns (\n
bool exists,\n            uint32 _key,\n            uint224 _value\n
)\n    {\n        uint256 pos = self._checkpoints.length;\n        if
(pos == 0) {\n            return (false, 0, 0);\n        } else {\n
Checkpoint224 memory ckpt = _unsafeAccess(self._checkpoints, pos - 1);\n
return (true, ckpt._key, ckpt._value);\n        }\n    }\n\n    /**\n
* @dev Returns the number of checkpoint.\n     */\n    function
length(Trace224 storage self) internal view returns (uint256) {\n
return self._checkpoints.length;\n    }\n\n    /**\n     * @dev Pushes a
(`key`, `value`) pair into an ordered list of checkpoints, either by
inserting a new checkpoint,\n     * or by updating the last one.\n
*/\n    function _insert(\n        Checkpoint224[] storage self,\n
uint32 key,\n        uint224 value\n    ) private returns (uint224,
uint224) {\n        uint256 pos = self.length;\n\n        if (pos > 0)
{\n            // Copying to memory is important here.\n
Checkpoint224 memory last = _unsafeAccess(self, pos - 1);\n\n
// Checkpoints keys must be increasing.\n            require(last._key <=
key, \"Checkpoint: invalid key\");\n\n            // Update or push new
checkpoint\n            if (last._key == key) {\n
_unsafeAccess(self, pos - 1)._value = value;\n            } else {\n
self.push(Checkpoint224({_key: key, _value: value}));\n            }\n
return (last._value, value);\n        } else {\n
self.push(Checkpoint224({_key: key, _value: value}));\n            return
(0, value);\n        }\n    }\n\n    /**\n     * @dev Return the index of
the oldest checkpoint whose key is greater than the search key, or `high`
if there is none.\n     * `low` and `high` define a section where to do
the search, with inclusive `low` and exclusive `high`.\n     *\n     *
WARNING: `high` should not be greater than the array's length.\n     */\n
function _upperBinaryLookup(\n        Checkpoint224[] storage self,\n
uint32 key,\n        uint256 low,\n        uint256 high\n    ) private
view returns (uint256) {\n        while (low < high) {\n
uint256 mid = MathUpgradeable.average(low, high);\n            if
(_unsafeAccess(self, mid)._key > key) {\n                high = mid;\n
} else {\n                low = mid + 1;\n            }\n        }\n
return high;\n    }\n\n    /**\n     * @dev Return the index of the
oldest checkpoint whose key is greater or equal than the search key, or
`high` if there is none.\n     * `low` and `high` define a section where
to do the search, with inclusive `low` and exclusive `high`.\n     *\n
* WARNING: `high` should not be greater than the array's length.\n
*/\n    function _lowerBinaryLookup(\n        Checkpoint224[] storage
self,\n        uint32 key,\n        uint256 low,\n        uint256 high\n
) private view returns (uint256) {\n        while (low < high) {\n
uint256 mid = MathUpgradeable.average(low, high);\n            if
(_unsafeAccess(self, mid)._key < key) {\n                low = mid + 1;\n
} else {\n                high = mid;\n            }\n        }\n
return high;\n    }\n\n    /**\n     * @dev Access an element of the
array without performing bounds check. The position is assumed to be
within bounds.\n     */\n    function _unsafeAccess(Checkpoint224[]
storage self, uint256 pos)\n        private\n        pure\n
returns (Checkpoint224 storage result)\n    {\n        assembly {\n
mstore(0, self.slot)\n            result.slot := add(keccak256(0, 0x20),
pos)\n        }\n    }\n\n    struct Trace160 {\n        Checkpoint160[]
_checkpoints;\n    }\n\n    struct Checkpoint160 {\n        uint96
_key;\n        uint160 _value;\n    }\n\n    /**\n     * @dev Pushes a
(`key`, `value`) pair into a Trace160 so that it is stored as the
checkpoint.\n     *\n     * Returns previous value and new value.\n
*/\n    function push(\n        Trace160 storage self,\n        uint96
key,\n        uint160 value\n    ) internal returns (uint160, uint160)
{\n        return _insert(self._checkpoints, key, value);\n    }\n\n
/**\n     * @dev Returns the value in the oldest checkpoint with key
greater or equal than the search key, or zero if there is none.\n
*/\n    function lowerLookup(Trace160 storage self, uint96 key) internal
view returns (uint160) {\n        uint256 len =
self._checkpoints.length;\n        uint256 pos =
_lowerBinaryLookup(self._checkpoints, key, 0, len);\n        return pos
== len ? 0 : _unsafeAccess(self._checkpoints, pos)._value;\n    }\n\n
/**\n     * @dev Returns the value in the most recent checkpoint with key
lower or equal than the search key.\n     */\n    function
upperLookup(Trace160 storage self, uint96 key) internal view returns
(uint160) {\n        uint256 len = self._checkpoints.length;\n
uint256 pos = _upperBinaryLookup(self._checkpoints, key, 0, len);\n
return pos == 0 ? 0 : _unsafeAccess(self._checkpoints, pos - 1)._value;\n
}\n\n    /**\n     * @dev Returns the value in the most recent
checkpoint, or zero if there are no checkpoints.\n     */\n    function
latest(Trace160 storage self) internal view returns (uint160) {\n
uint256 pos = self._checkpoints.length;\n        return pos == 0 ? 0 :
_unsafeAccess(self._checkpoints, pos - 1)._value;\n    }\n\n    /**\n
* @dev Returns whether there is a checkpoint in the structure (i.e. it is
not empty), and if so the key and value\n     * in the most recent
checkpoint.\n     */\n    function latestCheckpoint(Trace160 storage
self)\n        internal\n        view\n        returns (\n
bool exists,\n            uint96 _key,\n            uint160 _value\n
)\n    {\n        uint256 pos = self._checkpoints.length;\n        if
(pos == 0) {\n            return (false, 0, 0);\n        } else {\n
Checkpoint160 memory ckpt = _unsafeAccess(self._checkpoints, pos - 1);\n
return (true, ckpt._key, ckpt._value);\n        }\n    }\n\n    /**\n
* @dev Returns the number of checkpoint.\n     */\n    function
```

length(Trace160 storage self) internal view returns (uint256) {\n   return self._checkpoints.length;\n   }\n\n   /**\n   * @dev Pushes a (`key`, `value`) pair into an ordered list of checkpoints, either by inserting a new checkpoint,\n   * or by updating the last one.\n   */\n   function _insert(\n   Checkpoint160[] storage self,\n   uint96 key,\n   uint160 value\n   ) private returns (uint160, uint160) {\n   uint256 pos = self.length;\n\n   if (pos > 0) {\n   // Copying to memory is important here.\n   Checkpoint160 memory last = _unsafeAccess(self, pos - 1);\n\n   // Checkpoints keys must be increasing.\n   require(last._key <= key, \"Checkpoint: invalid key\");\n\n   // Update or push new checkpoint\n   if (last._key == key) {\n   _unsafeAccess(self, pos - 1)._value = value;\n   } else {\n   self.push(Checkpoint160({_key: key, _value: value}));\n   }\n   return (last._value, value);\n   } else {\n   self.push(Checkpoint160({_key: key, _value: value}));\n   return (0, value);\n   }\n   }\n\n   /**\n   * @dev Return the index of the oldest checkpoint whose key is greater than the search key, or `high` if there is none.\n   * `low` and `high` define a section where to do the search, with inclusive `low` and exclusive `high`.\n   *\n   * WARNING: `high` should not be greater than the array's length.\n   */\n   function _upperBinaryLookup(\n   Checkpoint160[] storage self,\n   uint96 key,\n   uint256 low,\n   uint256 high\n   ) private view returns (uint256) {\n   while (low < high) {\n   uint256 mid = MathUpgradeable.average(low, high);\n   if (_unsafeAccess(self, mid)._key > key) {\n   high = mid;\n   } else {\n   low = mid + 1;\n   }\n   }\n   return high;\n   }\n\n   /**\n   * @dev Return the index of the oldest checkpoint whose key is greater or equal than the search key, or `high` if there is none.\n   * `low` and `high` define a section where to do the search, with inclusive `low` and exclusive `high`.\n   *\n   * WARNING: `high` should not be greater than the array's length.\n   */\n   function _lowerBinaryLookup(\n   Checkpoint160[] storage self,\n   uint96 key,\n   uint256 low,\n   uint256 high\n   ) private view returns (uint256) {\n   while (low < high) {\n   uint256 mid = MathUpgradeable.average(low, high);\n   if (_unsafeAccess(self, mid)._key < key) {\n   low = mid + 1;\n   } else {\n   high = mid;\n   }\n   }\n   return high;\n   }\n\n   /**\n   * @dev Access an element of the array without performing bounds check. The position is assumed to be within bounds.\n   */\n   function _unsafeAccess(Checkpoint160[] storage self, uint256 pos)\n   private\n   pure\n   returns (Checkpoint160 storage result)\n   {\n   assembly {\n   mstore(0, self.slot)\n   result.slot := add(keccak256(0, 0x20), pos)\n   }\n   }\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts v4.4.1 (utils/Context.sol)\n\npragma solidity ^0.8.0;\nimport \"../proxy/utils/Initializable.sol\";\n\n/**\n * @dev Provides information about the current execution context, including the\n * sender of the transaction and its data. While these are generally available\n * via msg.sender and msg.data, they should not be accessed in such a direct\n * manner, since when dealing with meta-transactions the account

sending and\n * paying for execution may not be the actual sender (as far as an application\n * is concerned).\n *\n * This contract is only required for intermediate, library-like contracts.\n */\nabstract contract ContextUpgradeable is Initializable {\n   function __Context_init() internal onlyInitializing {\n   }\n\n   function __Context_init_unchained() internal onlyInitializing {\n   }\n   function _msgSender() internal view virtual returns (address) {\n   return msg.sender;\n   }\n\n   function _msgData() internal view virtual returns (bytes calldata) {\n   return msg.data;\n   }\n\n   /**\n   * @dev This empty reserved space is put in place to allow future versions to add new\n   * variables without shifting down storage in the inheritance chain.\n   * See https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n   */\n   uint256[50] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts v4.4.1 (utils/Counters.sol)\n\npragma solidity ^0.8.0;\n\n/**\n * @title Counters\n * @author Matt Condon (@shrugs)\n * @dev Provides counters that can only be incremented, decremented or reset. This can be used e.g. to track the number\n * of elements in a mapping, issuing ERC721 ids, or counting request ids.\n *\n * Include with `using Counters for Counters.Counter;`\n */\nlibrary CountersUpgradeable {\n   struct Counter {\n   // This variable should never be directly accessed by users of the library: interactions must be restricted to\n   // the library's function. As of Solidity v0.5.2, this cannot be enforced, though there is a proposal to add\n   // this feature: see https://github.com/ethereum/solidity/issues/4637\n   uint256 _value; // default: 0\n   }\n\n   function current(Counter storage counter) internal view returns (uint256) {\n   return counter._value;\n   }\n\n   function increment(Counter storage counter) internal {\n   unchecked {\n   counter._value += 1;\n   }\n   }\n\n   function decrement(Counter storage counter) internal {\n   uint256 value = counter._value;\n   require(value > 0, \"Counter: decrement overflow\");\n   unchecked {\n   counter._value = value - 1;\n   }\n   }\n\n   function reset(Counter storage counter) internal {\n   counter._value = 0;\n   }\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.8.0) (utils/Strings.sol)\n\npragma solidity ^0.8.0;\n\nimport \"./math/MathUpgradeable.sol\";\n\n/**\n * @dev String operations.\n */\nlibrary StringsUpgradeable {\n   bytes16 private constant _SYMBOLS = \"0123456789abcdef\";\n   uint8 private constant _ADDRESS_LENGTH = 20;\n\n   /**\n   * @dev Converts a `uint256` to its ASCII `string` decimal representation.\n   */\n   function toString(uint256 value) internal pure returns (string memory) {\n   unchecked {\n   uint256 length = MathUpgradeable.log10(value) + 1;\n   string memory buffer = new string(length);\n   uint256 ptr;\n   /// @solidity memory-safe-assembly\n   assembly {\n   ptr := add(buffer, add(32, length))\n   }\n   while (true) {\n   ptr--;\n   /// @solidity memory-safe-assembly\n   assembly {\n

mstore8(ptr, byte(mod(value, 10), _SYMBOLS))\n   }\n
value /= 10;\n   if (value == 0) break;\n   }\n
return buffer;\n   }\n   }\n\n   /**\n   * @dev Converts a
`uint256` to its ASCII `string` hexadecimal representation.\n   */\n
function toHexString(uint256 value) internal pure returns (string memory)
{\n   unchecked {\n   return toHexString(value,
MathUpgradeable.log256(value) + 1);\n   }\n   }\n\n   /**\n   *
@dev Converts a `uint256` to its ASCII `string` hexadecimal
representation with fixed length.\n   */\n   function
toHexString(uint256 value, uint256 length) internal pure returns (string
memory) {\n   bytes memory buffer = new bytes(2 * length + 2);\n
buffer[0] = \"0\";\n   buffer[1] = \"x\";\n   for (uint256 i =
2 * length + 1; i > 1; --i) {\n   buffer[i] = _SYMBOLS[value &
0xf];\n   value >>= 4;\n   }\n   require(value == 0,
\"Strings: hex length insufficient\");\n   return string(buffer);\n
}\n\n   /**\n   * @dev Converts an `address` with fixed length of 20
bytes to its not checksummed ASCII `string` hexadecimal representation.\n
*/\n   function toHexString(address addr) internal pure returns (string
memory) {\n   return toHexString(uint256(uint160(addr)),
_ADDRESS_LENGTH);\n   }\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/cryptography/ECDSAUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0) (utils/cryptography/ECDSA.sol)\n\npragma
solidity ^0.8.0;\n\nimport \"../StringsUpgradeable.sol\";\n\n/**\n * @dev
Elliptic Curve Digital Signature Algorithm (ECDSA) operations.\n *\n *
These functions can be used to verify that a message was signed by the
holder\n * of the private keys of a given address.\n */\nlibrary
ECDSAUpgradeable {\n   enum RecoverError {\n   NoError,\n
InvalidSignature,\n   InvalidSignatureLength,\n
InvalidSignatureS,\n   InvalidSignatureV // Deprecated in v4.8\n
}\n\n   function _throwError(RecoverError error) private pure {\n
if (error == RecoverError.NoError) {\n   return; // no error: do
nothing\n   } else if (error == RecoverError.InvalidSignature) {\n
revert(\"ECDSA: invalid signature\");\n   } else if (error ==
RecoverError.InvalidSignatureLength) {\n   revert(\"ECDSA:
invalid signature length\");\n   } else if (error ==
RecoverError.InvalidSignatureS) {\n   revert(\"ECDSA: invalid
signature 's' value\");\n   }\n   }\n\n   /**\n   * @dev Returns
the address that signed a hashed message (`hash`) with\n   *
`signature` or error string. This address can then be used for
verification purposes.\n   *\n   * The `ecrecover` EVM opcode allows
for malleable (non-unique) signatures:\n   * this function rejects them
by requiring the `s` value to be in the lower\n   * half order, and the
`v` value to be either 27 or 28.\n   *\n   * IMPORTANT: `hash` _must_
be the result of a hash operation for the\n   * verification to be
secure: it is possible to craft signatures that\n   * recover to
arbitrary addresses for non-hashed data. A safe way to ensure\n   *
this is by receiving a hash of the original message (which may
otherwise\n   * be too long), and then calling {toEthSignedMessageHash}
on it.\n   *\n   * Documentation for signature generation:\n   * -
with https://web3js.readthedocs.io/en/v1.3.4/web3-eth-
accounts.html#sign[Web3.js]\n   * - with

```
https://docs.ethers.io/v5/api/signer/#Signer-signMessage[ethers]\n
*\n     * _Available since v4.3._\n     */\n    function
tryRecover(bytes32 hash, bytes memory signature) internal pure returns
(address, RecoverError) {\n        if (signature.length == 65) {\n
bytes32 r;\n            bytes32 s;\n            uint8 v;\n            //
ecrecover takes the signature parameters, and the only way to get them\n
// currently is to use assembly.\n            /// @solidity memory-safe-
assembly\n            assembly {\n                r :=
mload(add(signature, 0x20))\n                s := mload(add(signature,
0x40))\n                v := byte(0, mload(add(signature, 0x60)))\n
}\n            return tryRecover(hash, v, r, s);\n        } else {\n
return (address(0), RecoverError.InvalidSignatureLength);\n        }\n
}\n\n    /**\n     * @dev Returns the address that signed a hashed
message (`hash`) with\n     * `signature`. This address can then be used
for verification purposes.\n     *\n     * The `ecrecover` EVM opcode
allows for malleable (non-unique) signatures:\n     * this function
rejects them by requiring the `s` value to be in the lower\n     * half
order, and the `v` value to be either 27 or 28.\n     *\n     *
IMPORTANT: `hash` _must_ be the result of a hash operation for the\n
* verification to be secure: it is possible to craft signatures that\n
* recover to arbitrary addresses for non-hashed data. A safe way to
ensure\n     * this is by receiving a hash of the original message (which
may otherwise\n     * be too long), and then calling
{toEthSignedMessageHash} on it.\n     */\n    function recover(bytes32
hash, bytes memory signature) internal pure returns (address) {\n
(address recovered, RecoverError error) = tryRecover(hash, signature);\n
_throwError(error);\n        return recovered;\n    }\n\n    /**\n     *
@dev Overload of {ECDSA-tryRecover} that receives the `r` and `vs` short-
signature fields separately.\n     *\n     * See
https://eips.ethereum.org/EIPS/eip-2098[EIP-2098 short signatures]\n
*\n     * _Available since v4.3._\n     */\n    function tryRecover(\n
bytes32 hash,\n        bytes32 r,\n        bytes32 vs\n    ) internal
pure returns (address, RecoverError) {\n        bytes32 s = vs &
bytes32(0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
f);\n        uint8 v = uint8((uint256(vs) >> 255) + 27);\n        return
tryRecover(hash, v, r, s);\n    }\n\n    /**\n     * @dev Overload of
{ECDSA-recover} that receives the `r and `vs` short-signature fields
separately.\n     *\n     * _Available since v4.2._\n     */\n
function recover(\n        bytes32 hash,\n        bytes32 r,\n
bytes32 vs\n    ) internal pure returns (address) {\n        (address
recovered, RecoverError error) = tryRecover(hash, r, vs);\n
_throwError(error);\n        return recovered;\n    }\n\n    /**\n     *
@dev Overload of {ECDSA-tryRecover} that receives the `v`,\n     * `r`
and `s` signature fields separately.\n     *\n     * _Available since
v4.3._\n     */\n    function tryRecover(\n        bytes32 hash,\n
uint8 v,\n        bytes32 r,\n        bytes32 s\n    ) internal pure
returns (address, RecoverError) {\n        // EIP-2 still allows
signature malleability for ecrecover(). Remove this possibility and make
the signature\n        // unique. Appendix F in the Ethereum Yellow paper
(https://ethereum.github.io/yellowpaper/paper.pdf), defines\n        //
the valid range for s in (301): 0 < s < secp256k1n ÷ 2 + 1, and for v in
(302): v ∈ {27, 28}. Most\n        // signatures from current libraries
generate a unique signature with an s-value in the lower half order.\n
//\n        // If your library generates malleable signatures, such as s-
values in the upper range, calculate a new s-value\n        // with
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141 - s1
and flip v from 27 to 28 or\n        // vice versa. If your library also
generates signatures with 0/1 for v instead 27/28, add 27 to v to
accept\n        // these malleable signatures as well.\n        if
(uint256(s) >
0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0) {\n
return (address(0), RecoverError.InvalidSignatureS);\n        }\n\n
// If the signature is valid (and not malleable), return the signer
address\n        address signer = ecrecover(hash, v, r, s);\n        if
(signer == address(0)) {\n            return (address(0),
RecoverError.InvalidSignature);\n        }\n\n        return (signer,
RecoverError.NoError);\n    }\n\n    /**\n     * @dev Overload of {ECDSA-
recover} that receives the `v`,\n     * `r` and `s` signature fields
separately.\n     */\n    function recover(\n        bytes32 hash,\n
uint8 v,\n        bytes32 r,\n        bytes32 s\n    ) internal pure
returns (address) {\n        (address recovered, RecoverError error) =
tryRecover(hash, v, r, s);\n        _throwError(error);\n        return
recovered;\n    }\n\n    /**\n     * @dev Returns an Ethereum Signed
Message, created from a `hash`. This\n     * produces hash corresponding
to the one signed with the\n     * https://eth.wiki/json-
rpc/API#eth_sign[`eth_sign`]\n     * JSON-RPC method as part of EIP-
191.\n     *\n     * See {recover}.\n     */\n    function
toEthSignedMessageHash(bytes32 hash) internal pure returns (bytes32) {\n
// 32 is the length in bytes of hash,\n        // enforced by the type
signature above\n        return
keccak256(abi.encodePacked(\"\\x19Ethereum Signed Message:\\n32\",
hash));\n    }\n\n    /**\n     * @dev Returns an Ethereum Signed
Message, created from `s`. This\n     * produces hash corresponding to
the one signed with the\n     * https://eth.wiki/json-
rpc/API#eth_sign[`eth_sign`]\n     * JSON-RPC method as part of EIP-
191.\n     *\n     * See {recover}.\n     */\n    function
toEthSignedMessageHash(bytes memory s) internal pure returns (bytes32)
{\n        return keccak256(abi.encodePacked(\"\\x19Ethereum Signed
Message:\\n\", StringsUpgradeable.toString(s.length), s));\n    }\n\n
/**\n     * @dev Returns an Ethereum Signed Typed Data, created from a\n
* `domainSeparator` and a `structHash`. This produces hash
corresponding\n     * to the one signed with the\n     *
https://eips.ethereum.org/EIPS/eip-712[`eth_signTypedData`]\n     * JSON-
RPC method as part of EIP-712.\n     *\n     * See {recover}.\n     */\n
function toTypedDataHash(bytes32 domainSeparator, bytes32 structHash)
internal pure returns (bytes32) {\n        return
keccak256(abi.encodePacked(\"\\x19\\x01\", domainSeparator,
structHash));\n    }\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/cryptography/EIP712Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0) (utils/cryptography/EIP712.sol)\n\npragma
solidity ^0.8.0;\n\nimport \"./ECDSAUpgradeable.sol\";\nimport
\"../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev
https://eips.ethereum.org/EIPS/eip-712[EIP 712] is a standard for hashing
and signing of typed structured data.\n *\n * The encoding specified in
the EIP is very generic, and such a generic implementation in Solidity is
```

```
not feasible,\n * thus this contract does not implement the encoding
itself. Protocols need to implement the type-specific encoding\n * they
need in their contracts using a combination of `abi.encode` and
`keccak256`.\n *\n * This contract implements the EIP 712 domain
separator ({_domainSeparatorV4}) that is used as part of the encoding\n *
scheme, and the final step of the encoding to obtain the message digest
that is then signed via ECDSA\n * ({_hashTypedDataV4}).\n *\n * The
implementation of the domain separator was designed to be as efficient as
possible while still properly updating\n * the chain id to protect
against replay attacks on an eventual fork of the chain.\n *\n * NOTE:
This contract implements the version of the encoding known as \"v4\", as
implemented by the JSON RPC method\n *
https://docs.metamask.io/guide/signing-data.html[`eth_signTypedDataV4` in
MetaMask].\n *\n * _Available since v3.4._\n *\n * @custom:storage-size
52\n */\nabstract contract EIP712Upgradeable is Initializable {\n    /*
solhint-disable var-name-mixedcase */\n    bytes32 private
_HASHED_NAME;\n    bytes32 private _HASHED_VERSION;\n    bytes32 private
constant _TYPE_HASH = keccak256(\"EIP712Domain(string name,string
version,uint256 chainId,address verifyingContract)\");\n\n    /* solhint-
enable var-name-mixedcase */\n\n    /**\n     * @dev Initializes the
domain separator and parameter caches.\n     *\n     * The meaning of
`name` and `version` is specified in\n     *
https://eips.ethereum.org/EIPS/eip-712#definition-of-domainseparator[EIP
712]:\n     *\n     * - `name`: the user readable name of the signing
domain, i.e. the name of the DApp or the protocol.\n     * - `version`:
the current major version of the signing domain.\n     *\n     * NOTE:
These parameters cannot be changed except through a
xref:learn::upgrading-smart-contracts.adoc[smart\n     * contract
upgrade].\n     */\n    function __EIP712_init(string memory name, string
memory version) internal onlyInitializing {\n
__EIP712_init_unchained(name, version);\n    }\n\n    function
__EIP712_init_unchained(string memory name, string memory version)
internal onlyInitializing {\n        bytes32 hashedName =
keccak256(bytes(name));\n        bytes32 hashedVersion =
keccak256(bytes(version));\n        _HASHED_NAME = hashedName;\n
_HASHED_VERSION = hashedVersion;\n    }\n\n    /**\n     * @dev Returns
the domain separator for the current chain.\n     */\n    function
_domainSeparatorV4() internal view returns (bytes32) {\n        return
_buildDomainSeparator(_TYPE_HASH, _EIP712NameHash(),
_EIP712VersionHash());\n    }\n\n    function _buildDomainSeparator(\n
bytes32 typeHash,\n        bytes32 nameHash,\n        bytes32
versionHash\n    ) private view returns (bytes32) {\n        return
keccak256(abi.encode(typeHash, nameHash, versionHash, block.chainid,
address(this)));\n    }\n\n    /**\n     * @dev Given an already
https://eips.ethereum.org/EIPS/eip-712#definition-of-hashstruct[hashed
struct], this\n     * function returns the hash of the fully encoded
EIP712 message for this domain.\n     *\n     * This hash can be used
together with {ECDSA-recover} to obtain the signer of a message. For
example:\n     *\n     * ```solidity\n     * bytes32 digest =
_hashTypedDataV4(keccak256(abi.encode(\n     *
keccak256(\"Mail(address to,string contents)\"),\n     *     mailTo,\n
*     keccak256(bytes(mailContents))\n     * )));\n     * address signer
= ECDSA.recover(digest, signature);\n     * ```\n     */\n    function
_hashTypedDataV4(bytes32 structHash) internal view virtual returns
```

(bytes32) {\n   return
ECDSAUpgradeable.toTypedDataHash(_domainSeparatorV4(), structHash);\n
}\n\n   /**\n   * @dev The hash of the name parameter for the EIP712
domain.\n   *\n   * NOTE: This function reads from storage by
default, but can be redefined to return a constant value if gas costs\n
* are a concern.\n   */\n   function _EIP712NameHash() internal
virtual view returns (bytes32) {\n   return _HASHED_NAME;\n   }\n\n
/**\n   * @dev The hash of the version parameter for the EIP712
domain.\n   *\n   * NOTE: This function reads from storage by
default, but can be redefined to return a constant value if gas costs\n
* are a concern.\n   */\n   function _EIP712VersionHash() internal
virtual view returns (bytes32) {\n   return _HASHED_VERSION;\n
}\n\n   /**\n   * @dev This empty reserved space is put in place to
allow future versions to add new\n   * variables without shifting down
storage in the inheritance chain.\n   * See
https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n
*/\n   uint256[50] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/introspection/ERC165Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts v4.4.1 (utils/introspection/ERC165.sol)\n\npragma solidity
^0.8.0;\n\nimport \"./IERC165Upgradeable.sol\";\nimport
\"../../proxy/utils/Initializable.sol\";\n\n/**\n * @dev Implementation
of the {IERC165} interface.\n *\n * Contracts that want to implement
ERC165 should inherit from this contract and override {supportsInterface}
to check\n * for the additional interface id that will be supported. For
example:\n *\n * ```solidity\n * function supportsInterface(bytes4
interfaceId) public view virtual override returns (bool) {\n *   return
interfaceId == type(MyInterface).interfaceId ||
super.supportsInterface(interfaceId);\n * }\n * ```\n *\n *
Alternatively, {ERC165Storage} provides an easier to use but more
expensive implementation.\n */\nabstract contract ERC165Upgradeable is
Initializable, IERC165Upgradeable {\n   function __ERC165_init()
internal onlyInitializing {\n   }\n\n   function
__ERC165_init_unchained() internal onlyInitializing {\n   }\n   /**\n
* @dev See {IERC165-supportsInterface}.\n   */\n   function
supportsInterface(bytes4 interfaceId) public view virtual override
returns (bool) {\n   return interfaceId ==
type(IERC165Upgradeable).interfaceId;\n   }\n\n   /**\n   * @dev This
empty reserved space is put in place to allow future versions to add
new\n   * variables without shifting down storage in the inheritance
chain.\n   * See
https://docs.openzeppelin.com/contracts/4.x/upgradeable#storage_gaps\n
*/\n   uint256[50] private __gap;\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/introspection/IERC165Upgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts v4.4.1 (utils/introspection/IERC165.sol)\n\npragma solidity
^0.8.0;\n\n/**\n * @dev Interface of the ERC165 standard, as defined in
the\n * https://eips.ethereum.org/EIPS/eip-165[EIP].\n *\n * Implementers
can declare support of contract interfaces, which can then be\n * queried
by others ({ERC165Checker}).\n *\n * For an implementation, see

{ERC165}.\n */\ninterface IERC165Upgradeable {\n   /**\n   * @dev
Returns true if this contract implements the interface defined by\n   *
`interfaceId`. See the corresponding\n   *
https://eips.ethereum.org/EIPS/eip-165#how-interfaces-are-identified[EIP
section]\n   * to learn more about how these ids are created.\n   *\n
* This function call must use less than 30 000 gas.\n   */\n
function supportsInterface(bytes4 interfaceId) external view returns
(bool);\n}\n"
    },
    "@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol":
{
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0) (utils/math/Math.sol)\n\npragma solidity
^0.8.0;\n\n/**\n * @dev Standard math utilities missing in the Solidity
language.\n */\nlibrary MathUpgradeable {\n   enum Rounding {\n
Down, // Toward negative infinity\n   Up, // Toward infinity\n
Zero // Toward zero\n   }\n\n   /**\n   * @dev Returns the largest of
two numbers.\n   */\n   function max(uint256 a, uint256 b) internal
pure returns (uint256) {\n   return a > b ? a : b;\n   }\n\n
/**\n   * @dev Returns the smallest of two numbers.\n   */\n
function min(uint256 a, uint256 b) internal pure returns (uint256) {\n
return a < b ? a : b;\n   }\n\n   /**\n   * @dev Returns the average
of two numbers. The result is rounded towards\n   * zero.\n   */\n
function average(uint256 a, uint256 b) internal pure returns (uint256)
{\n   // (a + b) / 2 can overflow.\n   return (a & b) + (a ^ b)
/ 2;\n   }\n\n   /**\n   * @dev Returns the ceiling of the division
of two numbers.\n   *\n   * This differs from standard division with
`/` in that it rounds up instead\n   * of rounding down.\n   */\n
function ceilDiv(uint256 a, uint256 b) internal pure returns (uint256)
{\n   // (a + b - 1) / b can overflow on addition, so we
distribute.\n   return a == 0 ? 0 : (a - 1) / b + 1;\n   }\n\n
/**\n   * @notice Calculates floor(x * y / denominator) with full
precision. Throws if result overflows a uint256 or denominator == 0\n
* @dev Original credit to Remco Bloemen under MIT license (https://xn--2-
umb.com/21/muldiv)\n   * with further edits by Uniswap Labs also under
MIT license.\n   */\n   function mulDiv(\n   uint256 x,\n
uint256 y,\n   uint256 denominator\n   ) internal pure returns
(uint256 result) {\n   unchecked {\n   // 512-bit multiply
[prod1 prod0] = x * y. Compute the product mod 2^256 and mod 2^256 - 1,
then use\n   // use the Chinese Remainder Theorem to reconstruct
the 512 bit result. The result is stored in two 256\n   //
variables such that product = prod1 * 2^256 + prod0.\n   uint256
prod0; // Least significant 256 bits of the product\n   uint256
prod1; // Most significant 256 bits of the product\n   assembly
{\n   let mm := mulmod(x, y, not(0))\n   prod0
:= mul(x, y)\n   prod1 := sub(sub(mm, prod0), lt(mm,
prod0))\n   }\n\n   // Handle non-overflow cases, 256
by 256 division.\n   if (prod1 == 0) {\n   return
prod0 / denominator;\n   }\n\n   // Make sure the
result is less than 2^256. Also prevents denominator == 0.\n
require(denominator > prod1);\n\n
///\n   // 512 by
256 division.\n
///\n\n   // Make

```
division exact by subtracting the remainder from [prod1 prod0].\n
uint256 remainder;\n            assembly {\n                // Compute
remainder using mulmod.\n                remainder := mulmod(x, y,
denominator)\n\n                // Subtract 256 bit number from 512 bit
number.\n                prod1 := sub(prod1, gt(remainder, prod0))\n
prod0 := sub(prod0, remainder)\n            }\n\n            // Factor
powers of two out of denominator and compute largest power of two divisor
of denominator. Always >= 1.\n            // See
https://cs.stackexchange.com/q/138556/92363.\n\n            // Does not
overflow because the denominator cannot be zero at this stage in the
function.\n            uint256 twos = denominator & (~denominator + 1);\n
assembly {\n                // Divide denominator by twos.\n
denominator := div(denominator, twos)\n\n                // Divide [prod1
prod0] by twos.\n                prod0 := div(prod0, twos)\n\n
// Flip twos such that it is 2^256 / twos. If twos is zero, then it
becomes one.\n                twos := add(div(sub(0, twos), twos), 1)\n
}\n\n            // Shift in bits from prod1 into prod0.\n
prod0 |= prod1 * twos;\n\n            // Invert denominator mod 2^256.
Now that denominator is an odd number, it has an inverse modulo 2^256
such\n            // that denominator * inv = 1 mod 2^256. Compute the
inverse by starting with a seed that is correct for\n            // four
bits. That is, denominator * inv = 1 mod 2^4.\n            uint256
inverse = (3 * denominator) ^ 2;\n\n            // Use the Newton-Raphson
iteration to improve the precision. Thanks to Hensel's lifting lemma,
this also works\n            // in modular arithmetic, doubling the
correct bits in each step.\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^8\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^16\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^32\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^64\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^128\n            inverse *= 2 - denominator *
inverse; // inverse mod 2^256\n\n            // Because the division is
now exact we can divide by multiplying with the modular inverse of
denominator.\n            // This will give us the correct result modulo
2^256. Since the preconditions guarantee that the outcome is\n
// less than 2^256, this is the final result. We don't need to compute
the high bits of the result and prod1\n            // is no longer
required.\n            result = prod0 * inverse;\n            return
result;\n        }\n    }\n\n    /**\n     * @notice Calculates x * y /
denominator with full precision, following the selected rounding
direction.\n     */\n    function mulDiv(\n        uint256 x,\n
uint256 y,\n        uint256 denominator,\n        Rounding rounding\n
) internal pure returns (uint256) {\n        uint256 result = mulDiv(x,
y, denominator);\n        if (rounding == Rounding.Up && mulmod(x, y,
denominator) > 0) {\n            result += 1;\n        }\n        return
result;\n    }\n\n    /**\n     * @dev Returns the square root of a
number. If the number is not a perfect square, the value is rounded
down.\n     *\n     * Inspired by Henry S. Warren, Jr.'s \"Hacker's
Delight\" (Chapter 11).\n     */\n    function sqrt(uint256 a) internal
pure returns (uint256) {\n        if (a == 0) {\n            return 0;\n
}\n\n        // For our first guess, we get the biggest power of 2 which
is smaller than the square root of the target.\n        //\n        // We
know that the \"msb\" (most significant bit) of our target number `a` is
a power of 2 such that we have\n        // `msb(a) <= a < 2*msb(a)`. This
```

```
value can be written `msb(a)=2**k` with `k=log2(a)`.\n        //\n
// This can be rewritten `2**log2(a) <= a < 2**(log2(a) + 1)`\n        //
→ `sqrt(2**k) <= sqrt(a) < sqrt(2**(k+1))`\n        // → `2**(k/2) <=
sqrt(a) < 2**((k+1)/2) <= 2**(k/2 + 1)`\n        //\n        //
Consequently, `2**(log2(a) / 2)` is a good first approximation of
`sqrt(a)` with at least 1 correct bit.\n        uint256 result = 1 <<
(log2(a) >> 1);\n\n        // At this point `result` is an estimation
with one bit of precision. We know the true value is a uint128,\n
// since it is the square root of a uint256. Newton's method converges
quadratically (precision doubles at\n        // every iteration). We thus
need at most 7 iteration to turn our partial result with one bit of
precision\n        // into the expected uint128 result.\n
unchecked {\n            result = (result + a / result) >> 1;\n
result = (result + a / result) >> 1;\n            result = (result + a /
result) >> 1;\n            result = (result + a / result) >> 1;\n
result = (result + a / result) >> 1;\n            result = (result + a /
result) >> 1;\n            result = (result + a / result) >> 1;\n
return min(result, a / result);\n        }\n    }\n\n    /**\n     *
@notice Calculates sqrt(a), following the selected rounding direction.\n
*/\n    function sqrt(uint256 a, Rounding rounding) internal pure returns
(uint256) {\n        unchecked {\n            uint256 result = sqrt(a);\n
return result + (rounding == Rounding.Up && result * result < a ? 1 :
0);\n        }\n    }\n\n    /**\n     * @dev Return the log in base 2,
rounded down, of a positive value.\n     * Returns 0 if given 0.\n
*/\n    function log2(uint256 value) internal pure returns (uint256) {\n
uint256 result = 0;\n        unchecked {\n            if (value >> 128 >
0) {\n                value >>= 128;\n                result += 128;\n
}\n            if (value >> 64 > 0) {\n                value >>= 64;\n
result += 64;\n            }\n            if (value >> 32 > 0) {\n
value >>= 32;\n                result += 32;\n            }\n
if (value >> 16 > 0) {\n                value >>= 16;\n
result += 16;\n            }\n            if (value >> 8 > 0) {\n
value >>= 8;\n                result += 8;\n            }\n            if
(value >> 4 > 0) {\n                value >>= 4;\n                result
+= 4;\n            }\n            if (value >> 2 > 0) {\n
value >>= 2;\n                result += 2;\n            }\n            if
(value >> 1 > 0) {\n                result += 1;\n            }\n
}\n        return result;\n    }\n\n    /**\n     * @dev Return the log
in base 2, following the selected rounding direction, of a positive
value.\n     * Returns 0 if given 0.\n     */\n    function log2(uint256
value, Rounding rounding) internal pure returns (uint256) {\n
unchecked {\n            uint256 result = log2(value);\n
return result + (rounding == Rounding.Up && 1 << result < value ? 1 :
0);\n        }\n    }\n\n    /**\n     * @dev Return the log in base 10,
rounded down, of a positive value.\n     * Returns 0 if given 0.\n
*/\n    function log10(uint256 value) internal pure returns (uint256) {\n
uint256 result = 0;\n        unchecked {\n            if (value >=
10**64) {\n                value /= 10**64;\n                result +=
64;\n            }\n            if (value >= 10**32) {\n
value /= 10**32;\n                result += 32;\n            }\n
if (value >= 10**16) {\n                value /= 10**16;\n
result += 16;\n            }\n            if (value >= 10**8) {\n
value /= 10**8;\n                result += 8;\n            }\n
if (value >= 10**4) {\n                value /= 10**4;\n
result += 4;\n            }\n            if (value >= 10**2) {\n
value /= 10**2;\n                result += 2;\n            }\n
if (value >= 10**1) {\n                result += 1;\n            }\n
}\n        return result;\n    }\n\n    /**\n     * @dev Return the log
in base 10, following the selected rounding direction, of a positive
value.\n     * Returns 0 if given 0.\n     */\n    function log10(uint256
value, Rounding rounding) internal pure returns (uint256) {\n
unchecked {\n            uint256 result = log10(value);\n
return result + (rounding == Rounding.Up && 10**result < value ? 1 :
0);\n        }\n    }\n\n    /**\n     * @dev Return the log in base 256,
rounded down, of a positive value.\n     * Returns 0 if given 0.\n
*\n     * Adding one to the result gives the number of pairs of hex
symbols needed to represent `value` as a hex string.\n     */\n
function log256(uint256 value) internal pure returns (uint256) {\n
uint256 result = 0;\n        unchecked {\n            if (value >> 128 >
0) {\n                value >>= 128;\n                result += 16;\n
}\n            if (value >> 64 > 0) {\n                value >>= 64;\n
result += 8;\n            }\n            if (value >> 32 > 0) {\n
value >>= 32;\n                result += 4;\n            }\n
if (value >> 16 > 0) {\n                value >>= 16;\n
result += 2;\n            }\n            if (value >> 8 > 0) {\n
result += 1;\n            }\n        }\n        return result;\n    }\n\n
/**\n     * @dev Return the log in base 10, following the selected
rounding direction, of a positive value.\n     * Returns 0 if given 0.\n
*/\n    function log256(uint256 value, Rounding rounding) internal pure
returns (uint256) {\n        unchecked {\n            uint256 result =
log256(value);\n            return result + (rounding == Rounding.Up && 1
<< (result * 8) < value ? 1 : 0);\n        }\n    }\n}\n"
    },
    "@openzeppelin/contracts-
upgradeable/utils/math/SafeCastUpgradeable.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin
Contracts (last updated v4.8.0) (utils/math/SafeCast.sol)\n// This file
was procedurally generated from
scripts/generate/templates/SafeCast.js.\n\npragma solidity
^0.8.0;\n\n/**\n * @dev Wrappers over Solidity's uintXX/intXX casting
operators with added overflow\n * checks.\n *\n * Downcasting from
uint256/int256 in Solidity does not revert on overflow. This can\n *
easily result in undesired exploitation or bugs, since developers
usually\n * assume that overflows raise errors. `SafeCast` restores this
intuition by\n * reverting the transaction when such an operation
overflows.\n *\n * Using this library instead of the unchecked operations
eliminates an entire\n * class of bugs, so it's recommended to use it
always.\n *\n * Can be combined with {SafeMath} and {SignedSafeMath} to
extend it to smaller types, by performing\n * all math on `uint256` and
`int256` and then downcasting.\n */\nlibrary SafeCastUpgradeable {\n
/**\n     * @dev Returns the downcasted uint248 from uint256, reverting
on\n     * overflow (when the input is greater than largest uint248).\n
*\n     * Counterpart to Solidity's `uint248` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 248 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint248(uint256 value)
internal pure returns (uint248) {\n        require(value <=
type(uint248).max, \"SafeCast: value doesn't fit in 248 bits\");\n
return uint248(value);\n    }\n\n    /**\n     * @dev Returns the
```

```
downcasted uint240 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint240).\n     *\n     * Counterpart to
Solidity's `uint240` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 240 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint240(uint256 value) internal pure
returns (uint240) {\n        require(value <= type(uint240).max,
\"SafeCast: value doesn't fit in 240 bits\");\n        return
uint240(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
uint232 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint232).\n     *\n     * Counterpart to Solidity's
`uint232` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 232 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint232(uint256 value) internal pure returns (uint232) {\n
require(value <= type(uint232).max, \"SafeCast: value doesn't fit in 232
bits\");\n        return uint232(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint224 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint224).\n     *\n
* Counterpart to Solidity's `uint224` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 224 bits\n     *\n
* _Available since v4.2._\n     */\n    function toUint224(uint256 value)
internal pure returns (uint224) {\n        require(value <=
type(uint224).max, \"SafeCast: value doesn't fit in 224 bits\");\n
return uint224(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint216 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint216).\n     *\n     * Counterpart to
Solidity's `uint216` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 216 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint216(uint256 value) internal pure
returns (uint216) {\n        require(value <= type(uint216).max,
\"SafeCast: value doesn't fit in 216 bits\");\n        return
uint216(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
uint208 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint208).\n     *\n     * Counterpart to Solidity's
`uint208` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 208 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint208(uint256 value) internal pure returns (uint208) {\n
require(value <= type(uint208).max, \"SafeCast: value doesn't fit in 208
bits\");\n        return uint208(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint200 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint200).\n     *\n
* Counterpart to Solidity's `uint200` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 200 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint200(uint256 value)
internal pure returns (uint200) {\n        require(value <=
type(uint200).max, \"SafeCast: value doesn't fit in 200 bits\");\n
return uint200(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint192 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint192).\n     *\n     * Counterpart to
Solidity's `uint192` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 192 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint192(uint256 value) internal pure
returns (uint192) {\n        require(value <= type(uint192).max,
\"SafeCast: value doesn't fit in 192 bits\");\n        return
uint192(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
```

```
uint184 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint184).\n     *\n     * Counterpart to Solidity's
`uint184` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 184 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint184(uint256 value) internal pure returns (uint184) {\n
require(value <= type(uint184).max, \"SafeCast: value doesn't fit in 184
bits\");\n        return uint184(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint176 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint176).\n     *\n
* Counterpart to Solidity's `uint176` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 176 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint176(uint256 value)
internal pure returns (uint176) {\n        require(value <=
type(uint176).max, \"SafeCast: value doesn't fit in 176 bits\");\n
return uint176(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint168 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint168).\n     *\n     * Counterpart to
Solidity's `uint168` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 168 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint168(uint256 value) internal pure
returns (uint168) {\n        require(value <= type(uint168).max,
\"SafeCast: value doesn't fit in 168 bits\");\n        return
uint168(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
uint160 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint160).\n     *\n     * Counterpart to Solidity's
`uint160` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 160 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint160(uint256 value) internal pure returns (uint160) {\n
require(value <= type(uint160).max, \"SafeCast: value doesn't fit in 160
bits\");\n        return uint160(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint152 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint152).\n     *\n
* Counterpart to Solidity's `uint152` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 152 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint152(uint256 value)
internal pure returns (uint152) {\n        require(value <=
type(uint152).max, \"SafeCast: value doesn't fit in 152 bits\");\n
return uint152(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint144 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint144).\n     *\n     * Counterpart to
Solidity's `uint144` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 144 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint144(uint256 value) internal pure
returns (uint144) {\n        require(value <= type(uint144).max,
\"SafeCast: value doesn't fit in 144 bits\");\n        return
uint144(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
uint136 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint136).\n     *\n     * Counterpart to Solidity's
`uint136` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 136 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint136(uint256 value) internal pure returns (uint136) {\n
require(value <= type(uint136).max, \"SafeCast: value doesn't fit in 136
bits\");\n        return uint136(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint128 from uint256, reverting on\n     *
```

```
overflow (when the input is greater than largest uint128).\n     *\n
* Counterpart to Solidity's `uint128` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 128 bits\n     *\n
* _Available since v2.5._\n     */\n    function toUint128(uint256 value)
internal pure returns (uint128) {\n        require(value <=
type(uint128).max, \"SafeCast: value doesn't fit in 128 bits\");\n
return uint128(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint120 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint120).\n     *\n     * Counterpart to
Solidity's `uint120` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 120 bits\n     *\n     * _Available since
v4.7._\n     */\n    function toUint120(uint256 value) internal pure
returns (uint120) {\n        require(value <= type(uint120).max,
\"SafeCast: value doesn't fit in 120 bits\");\n        return
uint120(value);\n    }\n\n    /**\n     * @dev Returns the downcasted
uint112 from uint256, reverting on\n     * overflow (when the input is
greater than largest uint112).\n     *\n     * Counterpart to Solidity's
`uint112` operator.\n     *\n     * Requirements:\n     *\n     * - input
must fit into 112 bits\n     *\n     * _Available since v4.7._\n     */\n
function toUint112(uint256 value) internal pure returns (uint112) {\n
require(value <= type(uint112).max, \"SafeCast: value doesn't fit in 112
bits\");\n        return uint112(value);\n    }\n\n    /**\n     * @dev
Returns the downcasted uint104 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint104).\n     *\n
* Counterpart to Solidity's `uint104` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 104 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint104(uint256 value)
internal pure returns (uint104) {\n        require(value <=
type(uint104).max, \"SafeCast: value doesn't fit in 104 bits\");\n
return uint104(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint96 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint96).\n     *\n     * Counterpart to
Solidity's `uint96` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 96 bits\n     *\n     * _Available since v4.2._\n
*/\n    function toUint96(uint256 value) internal pure returns (uint96)
{\n        require(value <= type(uint96).max, \"SafeCast: value doesn't
fit in 96 bits\");\n        return uint96(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint88 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint88).\n     *\n     *
Counterpart to Solidity's `uint88` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 88 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint88(uint256 value)
internal pure returns (uint88) {\n        require(value <=
type(uint88).max, \"SafeCast: value doesn't fit in 88 bits\");\n
return uint88(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint80 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint80).\n     *\n     * Counterpart to
Solidity's `uint80` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 80 bits\n     *\n     * _Available since v4.7._\n
*/\n    function toUint80(uint256 value) internal pure returns (uint80)
{\n        require(value <= type(uint80).max, \"SafeCast: value doesn't
fit in 80 bits\");\n        return uint80(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint72 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint72).\n     *\n     *
```

```
Counterpart to Solidity's `uint72` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 72 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint72(uint256 value)
internal pure returns (uint72) {\n        require(value <=
type(uint72).max, \"SafeCast: value doesn't fit in 72 bits\");\n
return uint72(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint64 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint64).\n     *\n     * Counterpart to
Solidity's `uint64` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 64 bits\n     *\n     * _Available since v2.5._\n
*/\n    function toUint64(uint256 value) internal pure returns (uint64)
{\n        require(value <= type(uint64).max, \"SafeCast: value doesn't
fit in 64 bits\");\n        return uint64(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint56 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint56).\n     *\n     *
Counterpart to Solidity's `uint56` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 56 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint56(uint256 value)
internal pure returns (uint56) {\n        require(value <=
type(uint56).max, \"SafeCast: value doesn't fit in 56 bits\");\n
return uint56(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint48 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint48).\n     *\n     * Counterpart to
Solidity's `uint48` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 48 bits\n     *\n     * _Available since v4.7._\n
*/\n    function toUint48(uint256 value) internal pure returns (uint48)
{\n        require(value <= type(uint48).max, \"SafeCast: value doesn't
fit in 48 bits\");\n        return uint48(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint40 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint40).\n     *\n     *
Counterpart to Solidity's `uint40` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 40 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint40(uint256 value)
internal pure returns (uint40) {\n        require(value <=
type(uint40).max, \"SafeCast: value doesn't fit in 40 bits\");\n
return uint40(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint32 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint32).\n     *\n     * Counterpart to
Solidity's `uint32` operator.\n     *\n     * Requirements:\n     *\n
* - input must fit into 32 bits\n     *\n     * _Available since v2.5._\n
*/\n    function toUint32(uint256 value) internal pure returns (uint32)
{\n        require(value <= type(uint32).max, \"SafeCast: value doesn't
fit in 32 bits\");\n        return uint32(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint24 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint24).\n     *\n     *
Counterpart to Solidity's `uint24` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 24 bits\n     *\n
* _Available since v4.7._\n     */\n    function toUint24(uint256 value)
internal pure returns (uint24) {\n        require(value <=
type(uint24).max, \"SafeCast: value doesn't fit in 24 bits\");\n
return uint24(value);\n    }\n\n    /**\n     * @dev Returns the
downcasted uint16 from uint256, reverting on\n     * overflow (when the
input is greater than largest uint16).\n     *\n     * Counterpart to
Solidity's `uint16` operator.\n     *\n     * Requirements:\n     *\n
```

```
* - input must fit into 16 bits\n     *\n     * _Available since v2.5._\n
*/\n    function toUint16(uint256 value) internal pure returns (uint16)
{\n        require(value <= type(uint16).max, \"SafeCast: value doesn't
fit in 16 bits\");\n        return uint16(value);\n    }\n\n    /**\n
* @dev Returns the downcasted uint8 from uint256, reverting on\n     *
overflow (when the input is greater than largest uint8).\n     *\n     *
Counterpart to Solidity's `uint8` operator.\n     *\n     *
Requirements:\n     *\n     * - input must fit into 8 bits\n     *\n
* _Available since v2.5._\n     */\n    function toUint8(uint256 value)
internal pure returns (uint8) {\n        require(value <=
type(uint8).max, \"SafeCast: value doesn't fit in 8 bits\");\n
return uint8(value);\n    }\n\n    /**\n     * @dev Converts a signed
int256 into an unsigned uint256.\n     *\n     * Requirements:\n     *\n
* - input must be greater than or equal to 0.\n     *\n     * _Available
since v3.0._\n     */\n    function toUint256(int256 value) internal pure
returns (uint256) {\n        require(value >= 0, \"SafeCast: value must
be positive\");\n        return uint256(value);\n    }\n\n    /**\n     *
@dev Returns the downcasted int248 from int256, reverting on\n     *
overflow (when the input is less than smallest int248 or\n     * greater
than largest int248).\n     *\n     * Counterpart to Solidity's `int248`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 248 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt248(int256 value) internal pure returns (int248 downcasted)
{\n        downcasted = int248(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 248 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int240 from int256, reverting on\n     *
overflow (when the input is less than smallest int240 or\n     * greater
than largest int240).\n     *\n     * Counterpart to Solidity's `int240`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 240 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt240(int256 value) internal pure returns (int240 downcasted)
{\n        downcasted = int240(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 240 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int232 from int256, reverting on\n     *
overflow (when the input is less than smallest int232 or\n     * greater
than largest int232).\n     *\n     * Counterpart to Solidity's `int232`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 232 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt232(int256 value) internal pure returns (int232 downcasted)
{\n        downcasted = int232(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 232 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int224 from int256, reverting on\n     *
overflow (when the input is less than smallest int224 or\n     * greater
than largest int224).\n     *\n     * Counterpart to Solidity's `int224`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 224 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt224(int256 value) internal pure returns (int224 downcasted)
{\n        downcasted = int224(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 224 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int216 from int256, reverting on\n     *
overflow (when the input is less than smallest int216 or\n     * greater
than largest int216).\n     *\n     * Counterpart to Solidity's `int216`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 216 bits\n     *\n     * _Available since v4.7._\n     */\n
```

```
function toInt216(int256 value) internal pure returns (int216 downcasted)
{\n        downcasted = int216(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 216 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int208 from int256, reverting on\n     *
overflow (when the input is less than smallest int208 or\n     * greater
than largest int208).\n     *\n     * Counterpart to Solidity's `int208`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 208 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt208(int256 value) internal pure returns (int208 downcasted)
{\n        downcasted = int208(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 208 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int200 from int256, reverting on\n     *
overflow (when the input is less than smallest int200 or\n     * greater
than largest int200).\n     *\n     * Counterpart to Solidity's `int200`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 200 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt200(int256 value) internal pure returns (int200 downcasted)
{\n        downcasted = int200(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 200 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int192 from int256, reverting on\n     *
overflow (when the input is less than smallest int192 or\n     * greater
than largest int192).\n     *\n     * Counterpart to Solidity's `int192`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 192 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt192(int256 value) internal pure returns (int192 downcasted)
{\n        downcasted = int192(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 192 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int184 from int256, reverting on\n     *
overflow (when the input is less than smallest int184 or\n     * greater
than largest int184).\n     *\n     * Counterpart to Solidity's `int184`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 184 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt184(int256 value) internal pure returns (int184 downcasted)
{\n        downcasted = int184(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 184 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int176 from int256, reverting on\n     *
overflow (when the input is less than smallest int176 or\n     * greater
than largest int176).\n     *\n     * Counterpart to Solidity's `int176`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 176 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt176(int256 value) internal pure returns (int176 downcasted)
{\n        downcasted = int176(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 176 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int168 from int256, reverting on\n     *
overflow (when the input is less than smallest int168 or\n     * greater
than largest int168).\n     *\n     * Counterpart to Solidity's `int168`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 168 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt168(int256 value) internal pure returns (int168 downcasted)
{\n        downcasted = int168(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 168 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int160 from int256, reverting on\n     *
overflow (when the input is less than smallest int160 or\n     * greater
than largest int160).\n     *\n     * Counterpart to Solidity's `int160`
```

operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 160 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt160(int256 value) internal pure returns (int160 downcasted)
{\n   downcasted = int160(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 160 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int152 from int256, reverting on\n   *
overflow (when the input is less than smallest int152 or\n   * greater
than largest int152).\n   *\n   * Counterpart to Solidity's `int152`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 152 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt152(int256 value) internal pure returns (int152 downcasted)
{\n   downcasted = int152(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 152 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int144 from int256, reverting on\n   *
overflow (when the input is less than smallest int144 or\n   * greater
than largest int144).\n   *\n   * Counterpart to Solidity's `int144`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 144 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt144(int256 value) internal pure returns (int144 downcasted)
{\n   downcasted = int144(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 144 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int136 from int256, reverting on\n   *
overflow (when the input is less than smallest int136 or\n   * greater
than largest int136).\n   *\n   * Counterpart to Solidity's `int136`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 136 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt136(int256 value) internal pure returns (int136 downcasted)
{\n   downcasted = int136(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 136 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int128 from int256, reverting on\n   *
overflow (when the input is less than smallest int128 or\n   * greater
than largest int128).\n   *\n   * Counterpart to Solidity's `int128`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 128 bits\n   *\n   * _Available since v3.1._\n   */\n
function toInt128(int256 value) internal pure returns (int128 downcasted)
{\n   downcasted = int128(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 128 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int120 from int256, reverting on\n   *
overflow (when the input is less than smallest int120 or\n   * greater
than largest int120).\n   *\n   * Counterpart to Solidity's `int120`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 120 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt120(int256 value) internal pure returns (int120 downcasted)
{\n   downcasted = int120(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 120 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int112 from int256, reverting on\n   *
overflow (when the input is less than smallest int112 or\n   * greater
than largest int112).\n   *\n   * Counterpart to Solidity's `int112`
operator.\n   *\n   * Requirements:\n   *\n   * - input must fit
into 112 bits\n   *\n   * _Available since v4.7._\n   */\n
function toInt112(int256 value) internal pure returns (int112 downcasted)
{\n   downcasted = int112(value);\n   require(downcasted ==
value, \"SafeCast: value doesn't fit in 112 bits\");\n   }\n\n   /**\n
* @dev Returns the downcasted int104 from int256, reverting on\n   *

```
overflow (when the input is less than smallest int104 or\n     * greater
than largest int104).\n     *\n     * Counterpart to Solidity's `int104`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 104 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt104(int256 value) internal pure returns (int104 downcasted)
{\n        downcasted = int104(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 104 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int96 from int256, reverting on\n     *
overflow (when the input is less than smallest int96 or\n     * greater
than largest int96).\n     *\n     * Counterpart to Solidity's `int96`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 96 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt96(int256 value) internal pure returns (int96 downcasted)
{\n        downcasted = int96(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 96 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int88 from int256, reverting on\n     *
overflow (when the input is less than smallest int88 or\n     * greater
than largest int88).\n     *\n     * Counterpart to Solidity's `int88`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 88 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt88(int256 value) internal pure returns (int88 downcasted)
{\n        downcasted = int88(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 88 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int80 from int256, reverting on\n     *
overflow (when the input is less than smallest int80 or\n     * greater
than largest int80).\n     *\n     * Counterpart to Solidity's `int80`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 80 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt80(int256 value) internal pure returns (int80 downcasted)
{\n        downcasted = int80(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 80 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int72 from int256, reverting on\n     *
overflow (when the input is less than smallest int72 or\n     * greater
than largest int72).\n     *\n     * Counterpart to Solidity's `int72`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 72 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt72(int256 value) internal pure returns (int72 downcasted)
{\n        downcasted = int72(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 72 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int64 from int256, reverting on\n     *
overflow (when the input is less than smallest int64 or\n     * greater
than largest int64).\n     *\n     * Counterpart to Solidity's `int64`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 64 bits\n     *\n     * _Available since v3.1._\n     */\n
function toInt64(int256 value) internal pure returns (int64 downcasted)
{\n        downcasted = int64(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 64 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int56 from int256, reverting on\n     *
overflow (when the input is less than smallest int56 or\n     * greater
than largest int56).\n     *\n     * Counterpart to Solidity's `int56`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 56 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt56(int256 value) internal pure returns (int56 downcasted)
{\n        downcasted = int56(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 56 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int48 from int256, reverting on\n     *
overflow (when the input is less than smallest int48 or\n     * greater
than largest int48).\n     *\n     * Counterpart to Solidity's `int48`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 48 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt48(int256 value) internal pure returns (int48 downcasted)
{\n        downcasted = int48(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 48 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int40 from int256, reverting on\n     *
overflow (when the input is less than smallest int40 or\n     * greater
than largest int40).\n     *\n     * Counterpart to Solidity's `int40`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 40 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt40(int256 value) internal pure returns (int40 downcasted)
{\n        downcasted = int40(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 40 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int32 from int256, reverting on\n     *
overflow (when the input is less than smallest int32 or\n     * greater
than largest int32).\n     *\n     * Counterpart to Solidity's `int32`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 32 bits\n     *\n     * _Available since v3.1._\n     */\n
function toInt32(int256 value) internal pure returns (int32 downcasted)
{\n        downcasted = int32(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 32 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int24 from int256, reverting on\n     *
overflow (when the input is less than smallest int24 or\n     * greater
than largest int24).\n     *\n     * Counterpart to Solidity's `int24`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 24 bits\n     *\n     * _Available since v4.7._\n     */\n
function toInt24(int256 value) internal pure returns (int24 downcasted)
{\n        downcasted = int24(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 24 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int16 from int256, reverting on\n     *
overflow (when the input is less than smallest int16 or\n     * greater
than largest int16).\n     *\n     * Counterpart to Solidity's `int16`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 16 bits\n     *\n     * _Available since v3.1._\n     */\n
function toInt16(int256 value) internal pure returns (int16 downcasted)
{\n        downcasted = int16(value);\n        require(downcasted ==
value, \"SafeCast: value doesn't fit in 16 bits\");\n    }\n\n    /**\n
* @dev Returns the downcasted int8 from int256, reverting on\n     *
overflow (when the input is less than smallest int8 or\n     * greater
than largest int8).\n     *\n     * Counterpart to Solidity's `int8`
operator.\n     *\n     * Requirements:\n     *\n     * - input must fit
into 8 bits\n     *\n     * _Available since v3.1._\n     */\n
function toInt8(int256 value) internal pure returns (int8 downcasted) {\n
downcasted = int8(value);\n        require(downcasted == value,
\"SafeCast: value doesn't fit in 8 bits\");\n    }\n\n    /**\n     *
@dev Converts an unsigned uint256 into a signed int256.\n     *\n     *
Requirements:\n     *\n     * - input must be less than or equal to
maxInt256.\n     *\n     * _Available since v3.0._\n     */\n    function
toInt256(uint256 value) internal pure returns (int256) {\n        //
Note: Unsafe cast below is okay because `type(int256).max` is guaranteed
```

to be positive\n   require(value <= uint256(type(int256).max), \"SafeCast: value doesn't fit in an int256\");\n   return int256(value);\n   }\n}\n"
    },
    "@openzeppelin/contracts/token/ERC20/IERC20.sol": {
      "content": "// SPDX-License-Identifier: MIT\n// OpenZeppelin Contracts (last updated v4.6.0) (token/ERC20/IERC20.sol)\n\npragma solidity ^0.8.0;\n\n/**\n * @dev Interface of the ERC20 standard as defined in the EIP.\n */\ninterface IERC20 {\n   /**\n   * @dev Emitted when `value` tokens are moved from one account (`from`) to\n   * another (`to`).\n   *\n   * Note that `value` may be zero.\n   */\n   event Transfer(address indexed from, address indexed to, uint256 value);\n\n   /**\n   * @dev Emitted when the allowance of a `spender` for an `owner` is set by\n   * a call to {approve}. `value` is the new allowance.\n   */\n   event Approval(address indexed owner, address indexed spender, uint256 value);\n\n   /**\n   * @dev Returns the amount of tokens in existence.\n   */\n   function totalSupply() external view returns (uint256);\n\n   /**\n   * @dev Returns the amount of tokens owned by `account`.\n   */\n   function balanceOf(address account) external view returns (uint256);\n\n   /**\n   * @dev Moves `amount` tokens from the caller's account to `to`.\n   *\n   * Returns a boolean value indicating whether the operation succeeded.\n   *\n   * Emits a {Transfer} event.\n   */\n   function transfer(address to, uint256 amount) external returns (bool);\n\n   /**\n   * @dev Returns the remaining number of tokens that `spender` will be\n   * allowed to spend on behalf of `owner` through {transferFrom}. This is\n   * zero by default.\n   *\n   * This value changes when {approve} or {transferFrom} are called.\n   */\n   function allowance(address owner, address spender) external view returns (uint256);\n\n   /**\n   * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.\n   *\n   * Returns a boolean value indicating whether the operation succeeded.\n   *\n   * IMPORTANT: Beware that changing an allowance with this method brings the risk\n   * that someone may use both the old and the new allowance by unfortunate\n   * transaction ordering. One possible solution to mitigate this race\n   * condition is to first reduce the spender's allowance to 0 and set the\n   * desired value afterwards:\n   * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729\n   *\n   * Emits an {Approval} event.\n   */\n   function approve(address spender, uint256 amount) external returns (bool);\n\n   /**\n   * @dev Moves `amount` tokens from `from` to `to` using the\n   * allowance mechanism. `amount` is then deducted from the caller's\n   * allowance.\n   *\n   * Returns a boolean value indicating whether the operation succeeded.\n   *\n   * Emits a {Transfer} event.\n   */\n   function transferFrom(\n   address from,\n   address to,\n   uint256 amount\n   ) external returns (bool);\n}\n"
    },
    "contracts/N2MCommonStorage.sol": {
      "content": "/** --- //\n *
//\n *   .:::.
//\n *   .:::::::.
//\n *   ::::::::.
//\n *   .:::::::::.

```
//\n *                             ..:::.              ..
//\n *                          .::::.                 ::::..
//\n *                      ..::::..                   ::::::::.
//\n *                   .::::.                        :::.  ..:::.
//\n *               ..::::..                          :::.      .:::.
//\n *            .::::.                               :::.
.:::..           //\n *         .::::..                   ..
:::.            .::::.        //\n *     .::::.              ..:::=-
::::               ..::::.    //\n *    :::.               .::::::::===:
::::::.               .::::   //\n *   .::.            .::::::::::::=====.
::::::::::.             .::.  //\n *   .::
.::::::::::::::=======.           :::::::::::::..         ::.  //\n *
.::        .::::::::::::::::=========-         ::::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==========:
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     :::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::===============.   :::::::::::::::::
::.  //\n *   .::        .::::::::::::::::================-.
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================:::::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::==================-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-===============-
::::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.=============-::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::     :============-:::::::::::::::        ::.  //\n *
.::        .::::::::::::::::       :==========-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-========-
::::::::::::::::        ::.  //\n *   .::         .::::::::::::::
.=======-:::::::::::::.         ::.  //\n *   .::.
.::::::::::            .=====-::::::::::..            ::.  //\n *
:::..              ..::::::              :===-::::::..              .:::.
//\n *      .::::..              .:::                -=-:::.
.::::.    //\n *         .::::.           .:::                 ..
.::::.       //\n *           .::::.         .:::
..:::.           //\n *               .:::.      .:::
.::::.              //\n *                .:::..  .:::
..:::..                 //\n *                  .::::.:::
.::::.                     //\n *                      ..::::
..:::..                        //\n *                          .:
.::::.                            //\n *
:::::.::::.                               //\n *
::::::::.                                  //\n *
:::::::.                                   //\n *
.::::.                                    //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
```

```
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/           //\n * ----------------
--------------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.18;\n\nimport
\"@nfts2me/contracts/interfaces/IN2M_ERCStorage.sol\";\nimport
\"@nfts2me/contracts/interfaces/IN2MCrossFactory.sol\";\nimport
\"@nfts2me/contracts/ownable/NFTOwnableUpgradeable.sol\";\nimport
{IOperatorFilterRegistry} from
\"@nfts2me/contracts/interfaces/IOperatorFilterRegistry.sol\";\nimport
\"./N2MVersion.sol\";\n\n/// @title NFTs2Me.com Smart Contracts\n///
@author The NFTs2Me Team\n/// @notice Read our terms of service\n///
@custom:security-contact security@nfts2me.com\n/// @custom:terms-of-
service https://nfts2me.com/terms-of-service/\n/// @custom:website
https://nfts2me.com/\nabstract contract N2MCommonStorage is\n
NFTOwnableUpgradeable,\n    IN2M_ERCStorage,\n    N2MVersion\n{\n    ///
CONSTANTS\n    address internal constant DEFAULT_SUBSCRIPTION =
address(0x3cc6CddA760b79bAfa08dF41ECFA224f810dCeB6);\n    address
internal constant OPENSEA_CONDUIT =
address(0x1E0049783F008A0085193E00003D00cd54003c71);\n    address
internal constant N2M_CONDUIT =
address(0x88899DC0B84C6E726840e00DFb94ABc6248825eC);\n
IOperatorFilterRegistry internal constant operatorFilterRegistry =
IOperatorFilterRegistry(0x000000000000AAeB6D7670E522A718067333cd4E);\n
address internal constant N2M_PRESALE_SIGNER =
address(0xC0ffee06CE3D6689305035601a055A96acd619c6);\n    address
internal constant N2M_TREASURY =
address(0x955aF4de9Ca03f84c9462457D075aCABf1A8AfC8);\n    uint256
internal constant N2M_FEE = 5_00;\n    uint256 internal constant
MAX_AFFILIATE_DISCOUNT = 100_00;\n    uint256 internal constant
MAX_AFFILIATE_PERCENTAGE = 100_00;\n    uint256 internal constant
NOT_ENTERED = 0;\n\n    /// IMMUTABLE    \n    address payable internal
immutable _factory;\n\n    bytes32 internal _baseURICIDHash;\n    bytes32
internal _placeholderImageCIDHash;\n    bytes32 internal
_contractURIMetadataCIDHash;\n\n    mapping(address => uint256) internal
_pendingAffiliateBalance;\n    uint256 internal
_pendingTotalAffiliatesBalance;\n\n    RevenueAddress[] internal
_revenueInfo;\n    mapping(address => AffiliateInformation) internal
_affiliatesInfo;\n\n    uint256 internal _mintPrice;\n    uint256
internal _withdrawnAmount;\n    uint256 internal _reentrancyEntered;\n
uint256 internal _dropDateTimestamp;\n    uint256 internal
_endDateTimestamp; \n\n    mapping(address => uint256) internal
_withdrawnERC20Amount;\n    address internal _erc20PaymentAddress;\n\n
mapping(address => RandomTicket) internal _randomTickets;\n
mapping(bytes => uint256) internal _usedAmountSignature;\n
mapping(uint256 => bool) internal _soulbound;\n    mapping(uint256 =>
bytes32) internal _customURICIDHashes;\n\n    uint32 internal
_soldTokens;\n    SalePhase internal _currentPhase;\n
OperatorFilterStatus internal _operatorFilterStatus;\n    MintingType
internal _mintingType;
\n    uint16 internal _royaltyFee;\n    uint16 internal _maxPerAddress;
\n    uint32 internal _collectionSize;\n    bool internal
_isERC20Payment;\n    bool internal _soulboundCollection;\n\n    ///
@custom:oz-upgrades-unsafe-allow constructor\n    constructor(address
payable factoryAddress) {\n        _factory = factoryAddress;\n
_disableInitializers();\n    }\n\n    /// @notice Returns the address of
the current collection owner.\n    function owner() public view
override(NFTOwnableUpgradeable) returns (address collectionOwner) {\n
try IN2MCrossFactory(_factory).ownerOf(uint256(uint160(address(this))))
returns (address ownerOf) {\n            return ownerOf;\n        } catch
{}\n    }\n\n    function _strictOwner() internal view
override(NFTOwnableUpgradeable) returns (address ownerStrictAddress) {\n
try
IN2MCrossFactory(_factory).strictOwnerOf(uint256(uint160(address(this))))
returns (address strictOwnerOf) {\n            return strictOwnerOf;\n
} catch {}\n    }\n\n    function _getN2MFeeAddress() internal view
override(NFTOwnableUpgradeable) returns (address) {\n\n        try
IN2MCrossFactory(_factory).getN2MTreasuryAddress() returns (address
n2mTreasuryAddress) {\n            return n2mTreasuryAddress;\n        }
catch {\n            return N2M_TREASURY;\n        }\n    }    \n\n
modifier nonReentrant() {\n        _nonReentrantBefore();\n        _;\n
_nonReentrantAfter();\n    }\n\n    function _nonReentrantBefore()
private {\n        if (_reentrancyEntered != 0) revert
ReentrancyGuard();\n        _reentrancyEntered = 1;\n    }\n\n
function _nonReentrantAfter() private {\n
delete(_reentrancyEntered);\n    }    \n\n    /// @notice Returns true if
the metadata is fixed and immutable. If the metadata hasn't been fixed
yet it will return false. Once fixed, it can't be changed by anyone.\n
function isMetadataFixed() public view override returns (bool) {\n
return (_baseURICIDHash != 0 || (_mintingType ==
MintingType.CUSTOM_URI));\n    }\n\n}\n"
    },
    "contracts/N2MERC721.sol": {
      "content": "/** --------------------------------------------------
------------------------ //\n *
//\n *                                       .:::.
//\n *                                    .:::::::.
//\n *                                    ::::::::.
//\n *                                 .:::::::::.
//\n *                             ..::::.            ..
//\n *                          .:::::.               ::::..
//\n *                      ..::::..                  ::::::::.
//\n *                   .::::.                       :::.  ..:::.
//\n *               ..::::..                         :::.      .:::.
//\n *            .:::::.                             :::.
.:::..           //\n *        .:::..              ..
:::.           .:::::.        //\n *     .::::.              ..:::=-
::::              ..::::.     //\n *    :::.              .:::::::===:
::::::.              .::::   //\n *   .::.         .:::::::::::=====.
::::::::::.           .::.  //\n *   .::
.::::::::::::::=======.         ::::::::::::..          ::.  //\n *
.::        .::::::::::::::::========-        ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==========:
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     ::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=============.   ::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==============-.
```

```
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::==================-::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-================-
::::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.==============-::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::     :============-::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::       :===========-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-=========-
::::::::::::::::        ::.  //\n *   .::         .::::::::::::::
.=======-::::::::::::::.         ::.  //\n *   .::.
.::::::::::           .=====-:::::::::::..            ::.  //\n *
::::..          ..:::::::              :===-:::::::..              .:::.
//\n *      .::::..              .:::                -=-:::.
.::::.    //\n *         .::::.           .:::                 ..
.::::.       //\n *            .::::.         .:::
..:::.           //\n *               .:::.      .:::
.::::.              //\n *                  .::::..  .:::
..::::..                //\n *                    .::::.:::
.::::.                     //\n *                     ..::::
..::::..                      //\n *                        .:
.::::.                          //\n *
:::::.::::.                           //\n *
::::::::.                              //\n *
::::::::.                              //\n *
.::::.                                //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/            //\n * ----------------
------------------------------------------------------------ */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.18;\n\nimport
\"@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/utils/cryptography/EIP712Upgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721VotesUpgradeable.sol\";\nimport
\"@openzeppelin/contracts/token/ERC20/IERC20.sol\";\n\nimport
\"./N2MTokenCommon.sol\";\n\n/// @title NFTs2Me.com Smart Contracts for
ERC-721.\n/// @author The NFTs2Me Team\n/// @notice Read our terms of
service\n/// @custom:security-contact security@nfts2me.com\n///
@custom:terms-of-service https://nfts2me.com/terms-of-service/\n///
@custom:website https://nfts2me.com/\ncontract N2MERC721 is\n
```

```
N2MTokenCommon,\n    ERC721Upgradeable,\n    EIP712Upgradeable,\n
ERC721VotesUpgradeable\n{\n\n    /// @notice To be called to create the
collection. Can only be called once.\n    function initialize(\n
string memory tokenName,\n        string memory tokenSymbol,\n
uint256 iMintPrice,\n        bytes32 baseURICIDHash,\n        bytes32
placeholderImageCIDHash,\n        RevenueAddress[] calldata
revenueAddresses,\n        address iErc20PaymentAddress,\n        uint32
iTotalSupply,\n        uint16 iRoyaltyFee,\n        bool
soulboundCollection,\n        MintingType iMintingType\n    ) public
payable override initializer {\n        __ERC721_init(tokenName,
tokenSymbol);\n\n        if (iTotalSupply == 0) revert
TotalSupplyMustBeGreaterThanZero();\n        if (baseURICIDHash != 0 &&
placeholderImageCIDHash != 0) revert
CantSetBaseURIAndPlaceholderAtTheSameTime();\n        if (iRoyaltyFee >
50_00) revert RoyaltyFeeTooHigh();\n\n        _collectionSize =
iTotalSupply;\n        if (baseURICIDHash == 0) {\n            if
(placeholderImageCIDHash == 0) {\n                if (iMintingType !=
MintingType.CUSTOM_URI)\n                    revert
NoBaseURINorPlaceholderSet();\n            } else {\n
_placeholderImageCIDHash = placeholderImageCIDHash;\n            }\n
} else {\n            _baseURICIDHash = baseURICIDHash;\n        }\n\n
_mintPrice = iMintPrice;\n        _royaltyFee = iRoyaltyFee;\n        if
(iMintingType != MintingType.SEQUENTIAL) {\n            _mintingType =
iMintingType;\n        }\n        if (iErc20PaymentAddress != address(0))
{\n            _isERC20Payment = true;\n            _erc20PaymentAddress
= iErc20PaymentAddress;\n        }\n        if (soulboundCollection ==
true) {\n            _soulboundCollection = true;\n        }\n\n
if (revenueAddresses.length > 0) {\n            uint256
revenuePercentageTotal;\n            for (uint256 i; i <
revenueAddresses.length; ) {\n                revenuePercentageTotal +=
revenueAddresses[i].percentage;\n                unchecked {\n
++i;\n                }\n            }\n            _revenueInfo =
revenueAddresses;\n            if (revenuePercentageTotal > 100_00 -
N2M_FEE) revert InvalidRevenuePercentage();\n        }\n\n    }\n\n
constructor(address libraryAddress, address payable factoryAddress)
N2MTokenCommon(libraryAddress, factoryAddress) {}\n\n    /// @notice A
distinct Uniform Resource Identifier (URI) for a given asset.\n    ///
@dev Throws if `_tokenId` is not a valid NFT. URIs are defined in RFC\n
///  3986. The URI may point to a JSON file that conforms to the
\"ERC721\n    ///  Metadata JSON Schema\".\n    function tokenURI(uint256
tokenId)\n        public\n        view\n        override(N2MTokenCommon,
ERC721Upgradeable)\n        returns (string memory)\n    {\n
_requireMinted(tokenId);\n        return
IN2MLibrary(address(this)).tokenURIImpl(tokenId);\n    }\n\n    function
_exists(uint256 tokenId)\n        internal\n        view\n
override(ERC721Upgradeable, N2MTokenCommon)\n        returns (bool)\n
{\n        return super._exists(tokenId);\n    }\n\n    function
_mint(address to, uint256 tokenId)\n        internal\n
override(ERC721Upgradeable, N2MTokenCommon)\n    {\n
super._mint(to, tokenId);\n    }\n\n    /// @notice A descriptive name
for a collection of NFTs in this contract\n    function name()\n
public\n        view\n        override(ERC721Upgradeable,
N2MTokenCommon)\n        returns (string memory)\n    {\n        return
super.name();\n    }\n\n    function _beforeTokenTransfer(\n
address from,\n        address to,\n        uint256 firstTokenId,\n
uint256 batchSize\n    ) internal override {\n        if (\n
from != address(0) &&\n            (_soulbound[firstTokenId] ||
_soulboundCollection)\n        ) revert
NonTransferrableSoulboundNFT();\n\n
super._beforeTokenTransfer(from, to, firstTokenId, batchSize);\n    }\n\n
function _afterTokenTransfer(\n        address from,\n        address
to,\n        uint256 firstTokenId,\n        uint256 batchSize\n    )
internal override(ERC721Upgradeable, ERC721VotesUpgradeable) {\n
super._afterTokenTransfer(from, to, firstTokenId, batchSize);\n\n
if (_maxPerAddress != 0) {\n\n            if (balanceOf(to) >
_maxPerAddress) revert MaxPerAddressExceeded();\n        }\n    }\n\n
function _burn(uint256 tokenId)\n        internal\n
override(ERC721Upgradeable)\n    {\n        super._burn(tokenId);\n
if (_customURICIDHashes[tokenId] != 0) {\n            delete
_customURICIDHashes[tokenId];\n        }        \n    }\n\n    ///
@notice Query if a contract implements an interface\n    /// @param
interfaceId The interface identifier, as specified in ERC-165\n    ///
@dev Interface identification is specified in ERC-165. This function uses
less than 30,000 gas.\n    /// @return `true` if the contract implements
`interfaceId` and `interfaceId` is not 0xffffffff, `false` otherwise\n
function supportsInterface(bytes4 interfaceId)\n        public\n
view\n        override(ERC721Upgradeable, IERC165Upgradeable)\n
returns (bool)\n    {\n        return (\n\n        interfaceId ==
type(IERC2981Upgradeable).interfaceId ||
super.supportsInterface(interfaceId));\n    }\n\n    /// @notice An
abbreviated name for NFTs in this contract\n    /// @return the
collection symbol\n    function symbol()\n        public\n        view\n
virtual\n        override(IN2M_ERCBase, ERC721Upgradeable)\n
returns (string memory)\n    {\n        return super.symbol();\n    }\n\n
/// @notice Count all NFTs assigned to an owner\n    /// @dev NFTs
assigned to the zero address are considered invalid, and this\n    ///
function throws for queries about the zero address.\n    /// @param owner
An address for whom to query the balance\n    /// @return balance The
number of NFTs owned by `owner`, possibly zero\n    function
balanceOf(address owner) public view override(ERC721Upgradeable,
N2MTokenCommon) returns (uint256 balance) {\n        balance =
super.balanceOf(owner);\n        if (_mintingType == MintingType.RANDOM)
{\n            balance += _randomTickets[owner].amount;\n        }\n
}\n\n    function _EIP712NameHash() internal virtual override view
returns (bytes32) {\n\n        return keccak256(\"NFTs2Me\");\n    }\n\n
function _EIP712VersionHash() internal virtual override view returns
(bytes32) {\n        return keccak256(\"1\");\n    }    \n\n    ///
@notice Enable or disable approval for a third party (\"operator\") to
manage\n    ///  all of `msg.sender`'s assets\n    /// @dev Emits the
ApprovalForAll event. The contract MUST allow\n    ///  multiple
operators per owner.\n    /// @param operator Address to add to the set
of authorized operators\n    /// @param approved True if the operator is
approved, false to revoke approval\n    function
setApprovalForAll(address operator, bool approved) public override
onlyAllowedOperatorApproval(operator) {\n
super.setApprovalForAll(operator, approved);\n    }\n\n    /// @notice
Change or reaffirm the approved address for an NFT\n    /// @dev The zero
address indicates there is no approved address.\n    ///  Throws unless
```

```
`msg.sender` is the current NFT owner, or an authorized\n    ///
operator of the current owner.\n    /// @param operator The new approved
NFT controller\n    /// @param tokenId The NFT to approve\n    function
approve(address operator, uint256 tokenId) public override
onlyAllowedOperatorApproval(operator) {\n        super.approve(operator,
tokenId);\n    }\n\n    /// @notice Query if an address is an authorized
operator for another address\n    /// @param owner The address that owns
the NFTs\n    /// @param operator The address that acts on behalf of the
owner\n    /// @return True if `operator` is an approved operator for
`owner`, false otherwise\n    function isApprovedForAll(address owner,
address operator)\n    public\n    view\n    virtual\n    override\n
returns (bool)\n    {\n\n        if (operator == N2M_CONDUIT) return
true;\n        if (operator == OPENSEA_CONDUIT) return true;\n\n
return super.isApprovedForAll(owner, operator);\n    }\n\n    /// @notice
Transfer ownership of an NFT -- THE CALLER IS RESPONSIBLE\n    ///  TO
CONFIRM THAT `to` IS CAPABLE OF RECEIVING NFTS OR ELSE\n    ///  THEY MAY
BE PERMANENTLY LOST\n    /// @dev Throws unless `msg.sender` is the
current owner, an authorized\n    ///  operator, or the approved address
for this NFT. Throws if `from` is\n    ///  not the current owner. Throws
if `to` is the zero address. Throws if\n    ///  `tokenId` is not a valid
NFT.\n    /// @param from The current owner of the NFT\n    /// @param to
The new owner\n    /// @param tokenId The NFT to transfer\n    function
transferFrom(\n        address from,\n        address to,\n
uint256 tokenId\n    ) public override onlyAllowedOperator() {\n
super.transferFrom(from, to, tokenId);\n    }\n\n    /// @notice
Transfers the ownership of an NFT from one address to another address\n
/// @dev This works identically to the other function with an extra data
parameter,\n    ///  except this function just sets data to \"\".\n
/// @param from The current owner of the NFT\n    /// @param to The new
owner\n    /// @param tokenId The NFT to transfer\n    function
safeTransferFrom(\n        address from,\n        address to,\n
uint256 tokenId\n    ) public override onlyAllowedOperator() {\n
super.safeTransferFrom(from, to, tokenId);\n    }\n\n    /// @notice
Transfers the ownership of an NFT from one address to another address\n
/// @dev Throws unless `msg.sender` is the current owner, an authorized\n
///  operator, or the approved address for this NFT. Throws if `from`
is\n    ///  not the current owner. Throws if `to` is the zero address.
Throws if\n    ///  `tokenId` is not a valid NFT. When transfer is
complete, this function\n    ///  checks if `to` is a smart contract
(code size > 0). If so, it calls\n    ///  `onERC721Received` on `to` and
throws if the return value is not\n    ///
`bytes4(keccak256(\"onERC721Received(address,address,uint256,bytes)\"))`.
\n    /// @param from The current owner of the NFT\n    /// @param to The
new owner\n    /// @param tokenId The NFT to transfer\n    /// @param
data Additional data with no specified format, sent in call to `to`\n
function safeTransferFrom(\n        address from,\n        address to,\n
uint256 tokenId,\n        bytes memory data\n    ) public override
onlyAllowedOperator() {\n        super.safeTransferFrom(from, to,
tokenId, data);\n    }\n\n}\n"
    },
    "contracts/N2MTokenCommon.sol": {
      "content": "/** ----------------------------------------------------
------------------------ //\n *
//\n *                                   .:::.
```

```
//\n *                                    .:::::::.
//\n *                                    ::::::::.
//\n *                                 .::::::::::.
//\n *                             ..:::.                    ..
//\n *                          .::::.                 ::::..
//\n *                      ..::::..                   ::::::::.
//\n *                   .::::.                        :::.  ..::::.
//\n *               ..::::..                          :::.      .:::.
//\n *            .::::.                               :::.
.:::..           //\n *         .::::..              ..
:::.            .::::.        //\n *     .::::.               ..:::=-
::::               ..::::.    //\n *    :::.               .:::::::===:
::::::.               .::::   //\n *   .::.            .:::::::::::=====.
::::::::::.             .::.  //\n *   .::
.::::::::::::::=======.           ::::::::::::..           ::.  //\n *
.::        .::::::::::::::::=========-         :::::::::::::::::
::.  //\n *   .::        .::::::::::::::::==========:
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     :::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   :::::::::::::::::
::.  //\n *   .::        .::::::::::::::::===============-.
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================:::::::::::::::::         ::.  //\n *
.::        .::::::::::::::::==================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::==================-:::::::::::::::         ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-================-
::::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.==============-:::::::::::::::         ::.  //\n *   .::
.::::::::::::::::     :============-:::::::::::::::         ::.  //\n *
.::        .::::::::::::::::       :==========-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-========-
::::::::::::::::        ::.  //\n *   .::          .::::::::::::::
.=======-:::::::::::::.          ::.  //\n *   .::.
.::::::::::            .=====-::::::::::..            ::.  //\n *
:::..              ..::::::              :===-:::::::..             .:::.
//\n *      .::::..              .:::                -=-:::.
.::::.    //\n *         .::::.            .:::                 ..
.::::.       //\n *            .::::.         .:::
..:::.           //\n *                .:::.      .:::
.::::.              //\n *                   .::::..  .:::
..:::..                 //\n *                      .::::.:::
.::::.                     //\n *                         ..::::
..:::..                        //\n *                             .:
.::::.                      //\n *
:::::.::::.                              //\n *
::::::::.                                 //\n *
:::::::.                                   //\n *
.::::.                                    //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
```

```
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service: ⅝            //\n * --
--------------------------------------------------------------------------
-- */\n\n/// SPDX-License-Identifier: UNLICENSED\npragma solidity
^0.8.18;\n\nimport \"@openzeppelin/contracts-
upgradeable/token/ERC721/ERC721Upgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/utils/cryptography/EIP712Upgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721VotesUpgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/token/ERC721/extensions/ERC721URIStorageUpgradeable.sol\";\ni
mport \"@openzeppelin/contracts-
upgradeable/proxy/utils/Initializable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/interfaces/IERC2981Upgradeable.sol\";\nimport
\"@openzeppelin/contracts-
upgradeable/utils/cryptography/ECDSAUpgradeable.sol\";\n\n///
Utils\nimport \"@openzeppelin/contracts-
upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol\";\nimport
\"@openzeppelin/contracts/token/ERC20/IERC20.sol\";\n\nimport
\"@nfts2me/contracts/interfaces/IN2MCrossFactory.sol\";\nimport
\"@nfts2me/contracts/interfaces/IN2M_ERCBase.sol\";\nimport
\"./N2MCommonStorage.sol\";\n\ninterface IN2MLibrary {\n    function
tokenURIImpl(uint256 tokenId) external view returns (string
memory);\n}\n\n/// @title NFTs2Me.com Smart Contracts\n/// @author The
NFTs2Me Team\n/// @notice Read our terms of service\n///
@custom:security-contact security@nfts2me.com\n/// @custom:terms-of-
service https://nfts2me.com/terms-of-service/\n/// @custom:website
https://nfts2me.com/\nabstract contract N2MTokenCommon is
N2MCommonStorage, IN2M_ERCBase {\n    /// IMMUTABLE\n    address internal
immutable LIBRARY_ADDRESS;\n\n    /// @custom:oz-upgrades-unsafe-allow
constructor\n    constructor(address libraryAddress, address payable
factoryAddress) N2MCommonStorage(factoryAddress) {\n
LIBRARY_ADDRESS = libraryAddress;\n    }\n\n    function _mint(address
to, uint256 tokenId) internal virtual;\n\n    function _exists(uint256
tokenId) internal view virtual returns (bool);\n\n    function name()
external view virtual override returns (string memory);\n\n    function
tokenURI(uint256 tokenId) external view virtual override returns (string
memory);\n\n    function balanceOf(address owner) public view virtual
returns (uint256 balance);    \n\n    /// @notice Mints one NFT to the
caller (msg.sender). Requires `minting type` to be `sequential` and the
`mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment).\n    function mint() external payable override {\n
_requirePayment(_mintPrice, 1);\n        _checkPhase();\n        if
(_mintingType != MintingType.SEQUENTIAL) revert InvalidMintingType();\n
unchecked {\n            if ((++_soldTokens) > _collectionSize) revert
CollectionSoldOut();\n        }\n        _mint(msg.sender,
_soldTokens);\n    }\n\n    /// @notice Mints `amount` NFTs to the caller
(msg.sender). Requires `minting type` to be `sequential` and the
```

```
`mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment).\n    /// @param amount The number of NFTs to mint\n    function
mint(uint256 amount) external payable override {\n
_requirePayment(_mintPrice, amount);\n
_mintSequentialWithChecks(msg.sender, amount);\n    }\n\n    /// @notice
Mints `amount` NFTs to the caller (msg.sender) with a given `affiliate`.
Requires `minting type` to be `sequential` and the `mintPrice` to be send
(if `Native payment`) or approved (if `ERC-20` payment).\n    /// @param
amount The number of NFTs to mint\n    /// @param affiliate The affiliate
address\n    function mint(uint256 amount, address affiliate) external
payable override {\n        _requirePaymentWithAffiliates(amount,
affiliate);\n        _mintSequentialWithChecks(msg.sender, amount);\n
}\n\n    /// @notice Mints `amount` NFTs to `to` address. Requires
`minting type` to be `sequential` and the `mintPrice` to be send (if
`Native payment`) or approved (if `ERC-20` payment).\n    /// @param to
The address of the NFTs receiver\n    /// @param amount The number of
NFTs to mint    \n    function mintTo(address to, uint256 amount)
external payable override {\n        _requirePayment(_mintPrice,
amount);\n        _mintSequentialWithChecks(to, amount);\n    }\n\n
/// @notice Mints `amount` NFTs to `to` address with a given `affiliate`.
Requires `minting type` to be `sequential` and the `mintPrice` to be send
(if `Native payment`) or approved (if `ERC-20` payment).\n    /// @param
to The address of the NFTs receiver\n    /// @param amount The number of
NFTs to mint    \n    /// @param affiliate The affiliate address\n
function mintTo(address to, uint256 amount, address affiliate) external
payable override {\n        _requirePaymentWithAffiliates(amount,
affiliate);\n        _mintSequentialWithChecks(to, amount);\n    }\n\n
function _mintSequentialWithChecks(address to, uint256 amount) private
{\n        _checkPhase();\n        if (_mintingType !=
MintingType.SEQUENTIAL) revert InvalidMintingType();\n        if
((_soldTokens + amount) > _collectionSize) revert
CollectionSoldOut();\n\n        _mintSequential(to, amount);\n    }\n\n
function _mintSequential(address to, uint256 amount, bool soulbound)
private {\n        for (uint256 i; i < amount; ) {\n            unchecked
{\n                _mint(to, ++_soldTokens);\n            }\n
if (soulbound) _soulbound[_soldTokens] = true;\n            unchecked {\n
++i;\n            }\n        }\n    }\n\n    function
_mintSequential(address to, uint256 amount) internal virtual {\n
for (uint256 i; i < amount; ) {\n            unchecked {\n
_mint(to, ++_soldTokens);\n                ++i;\n            }\n
}\n    }\n\n    /// @notice Two phases on-chain random minting. Mints
`amount` NFTs tickets to `to` address. Requires `minting type` to be
`random` and the `mintPrice` to be send (if `Native payment`) or approved
(if `ERC-20` payment). Once minted, those tickets must be redeemed for an
actual token calling `redeemRandom()`.\n    /// @param to The address of
the NFTs receiver\n    /// @param amount The number of NFTs to mint    \n
function mintRandomTo(address to, uint256 amount) external payable
override {\n        _requirePayment(_mintPrice, amount);\n
_mintRandomWithChecks(to, amount);\n    }\n\n    /// @notice Two phases
on-chain random minting. Mints `amount` NFTs tickets to `to` address with
a given `affiliate`. Requires `minting type` to be `random` and the
`mintPrice` to be send (if `Native payment`) or approved (if `ERC-20`
payment). Once minted, those tickets must be redeemed for an actual token
calling `redeemRandom()`.\n    /// @param to The address of the NFTs
```

```
receiver\n    /// @param amount The number of NFTs to mint    \n    ///
@param affiliate The affiliate address\n    function mintRandomTo(address
to, uint256 amount, address affiliate) external payable override {\n
_requirePaymentWithAffiliates(amount, affiliate);\n
_mintRandomWithChecks(to, amount);\n    }\n\n    function
_mintRandomWithChecks(address to, uint256 amount) private {\n
_checkPhase();\n        if (_mintingType != MintingType.RANDOM) revert
InvalidMintingType();\n        if (_soldTokens + (amount) >
_collectionSize) revert CollectionSoldOut();\n\n        unchecked {\n
_randomTickets[to].blockNumberToReveal = block.number + 2;\n
_randomTickets[to].amount += amount;\n            _soldTokens +=
uint32(amount);\n        }\n\n        if (_maxPerAddress != 0) {\n\n
if (balanceOf(to) > _maxPerAddress) revert MaxPerAddressExceeded();\n
}        \n    }\n\n    /// @notice Redeems remaining random tickets
generated from `msg.sender` by calling `mintRandomTo` for actual NFTs.\n
function redeemRandom() external payable override {\n        uint256
blockNumberToReveal = _randomTickets[msg.sender].blockNumberToReveal;\n
uint256 amountToRedeem = _randomTickets[msg.sender].amount;\n\n        if
(amountToRedeem == 0) revert NothingToRedeem();\n        if (block.number
<= _randomTickets[msg.sender].blockNumberToReveal) revert
CantRevealYetWaitABitToBeAbleToRedeem();\n\n        bytes32
seedFromBlockNumber = blockhash(blockNumberToReveal);\n\n        if
(seedFromBlockNumber == 0) {\n\n            uint256 newBlockNumber =
((block.number & uint256(int256(-0x100))) + (blockNumberToReveal &
0xff));\n\n            if ((newBlockNumber >= block.number)) {\n
newBlockNumber -= 256;\n\n            }\n            seedFromBlockNumber
= blockhash(newBlockNumber);\n\n        }\n\n
delete(_randomTickets[msg.sender].blockNumberToReveal);\n
delete(_randomTickets[msg.sender].amount);\n\n        uint16
maxPerAddressTemp = _maxPerAddress;\n        delete(_maxPerAddress);\n
_mintRandom(msg.sender, amountToRedeem, seedFromBlockNumber, false);\n
_maxPerAddress = maxPerAddressTemp;\n    }\n\n    function
_mintRandom(address to, uint256 amount, bytes32 seed, bool soulbound)
private {\n        for (; amount > 0; ) {\n            uint256 tokenId =
_randomTokenId(seed, amount);\n            _mint(to, tokenId);\n
if (soulbound) _soulbound[tokenId] = true;\n            unchecked {\n
--amount;\n            }\n        }\n    }\n\n    function
_randomTokenId(bytes32 seed, uint256 extraModifier) private view returns
(uint256 tokenId) {\n\n        tokenId =
(uint256(keccak256(abi.encodePacked(seed, extraModifier))) %
_collectionSize) + 1;\n\n        while (_exists(tokenId)) {\n
unchecked {\n                tokenId = (tokenId % _collectionSize) + 1;\n
}\n        }\n    }\n\n    /// @notice Mints `amount` NFTs to `to`
address. Requires `minting type` to be `specify` and the `mintPrice` to
be send (if `Native payment`) or approved (if `ERC-20` payment).\n    ///
@param to The address of the NFTs receiver\n    /// @param tokenIds An
array of the specified tokens. They must not be minted, otherwise, it
will revert.\n    function mintSpecifyTo(address to, uint256[] memory
tokenIds)\n        external\n        payable\n        override\n    {\n
_requirePayment(_mintPrice, tokenIds.length);\n
_mintSpecifyWithChecks(to, tokenIds);\n    }\n\n    /// @notice Mints
`amount` NFTs to `to` address with a given `affiliate`. Requires `minting
type` to be `specify` and the `mintPrice` to be send (if `Native
payment`) or approved (if `ERC-20` payment).\n    /// @param to The
```

```
address of the NFTs receiver\n    /// @param tokenIds An array of the
specified tokens. They must not be minted, otherwise, it will revert.\n
/// @param affiliate The affiliate address\n    function
mintSpecifyTo(address to, uint256[] memory tokenIds, address affiliate)
\n        external\n        payable\n        override\n    {\n
_requirePaymentWithAffiliates(tokenIds.length, affiliate);\n
_mintSpecifyWithChecks(to, tokenIds);\n    }\n\n    function
_mintSpecifyWithChecks(address to, uint256[] memory tokenIds)\n
private\n    {\n        _checkPhase();\n        if (_mintingType !=
MintingType.SPECIFY) revert InvalidMintingType();\n\n        if
(_soldTokens + (tokenIds.length) > _collectionSize) revert
CollectionSoldOut();\n\n        _mintSpecify(to, tokenIds);\n    }\n\n
function _mintSpecify(\n        address to,\n        uint256[] memory
tokenIds,\n        bool soulbound\n    ) private {\n
_mintSpecify(to, tokenIds);\n        uint256 inputLength =
tokenIds.length;        \n        if (soulbound) {\n            for
(uint256 i; i < inputLength; ) {\n                _soulbound[tokenIds[i]]
= true;\n                unchecked {\n                    ++i;\n
}\n            }\n        }\n    }\n\n    function _mintSpecify(address
to, uint256[] memory tokenIds)\n        internal\n        virtual\n
{\n\n        uint256 inputLength = tokenIds.length;\n        unchecked
{\n            _soldTokens += uint32(inputLength);\n        }\n
for (uint256 i; i < inputLength; ) {\n            uint256 tokenId =
tokenIds[i];\n\n            if (tokenId == 0 || tokenId >
_collectionSize) revert InvalidTokenId();\n            _mint(to,
tokenId);\n            unchecked {\n                ++i;\n            }\n
}\n    }\n\n    /// @notice Mints one NFT to `to` address. Requires
`minting type` to be `customURI`.\n    /// @param to The address of the
NFTs receiver\n    /// @param customURICIDHash The CID of the given
token.\n    /// @param soulbound True if the NFT is a Soulbound Token
(SBT). If set, it can't be transferred.\n    function
mintCustomURITo(address to, bytes32 customURICIDHash, bool soulbound)\n
external\n        payable\n        override\n    {\n
_requirePayment(_mintPrice, 1);\n        _checkPhase();\n        if
(_mintingType != MintingType.CUSTOM_URI) revert InvalidMintingType();\n
unchecked {\n            if ((++_soldTokens) > _collectionSize) revert
CollectionSoldOut();\n        }\n        _mint(to, _soldTokens);\n\n
unchecked {\n            if (soulbound) _soulbound[_soldTokens] = true;\n
}\n        _customURICIDHashes[_soldTokens] = customURICIDHash;\n
}\n\n    /// @notice Only owner can call this function. Free of charge.
Mints sizeof(`to`) to `to` addresses. Requires `minting type` to be
`sequential`.\n    /// @param to The addresses of the NFTs receivers\n
/// @param soulbound True if the NFT is a Soulbound Token (SBT). If set,
it can't be transferred.\n    function airdropSequential(address[] memory
to, bool soulbound)\n        external\n        payable\n
override\n        onlyStrictOwner\n    {\n        if (_mintingType !=
MintingType.SEQUENTIAL) revert InvalidMintingType();\n        if
(_soldTokens + (to.length) > _collectionSize) revert
CollectionSoldOut();\n\n        uint256 toLength = to.length;\n
for (uint256 i; i < toLength; ) {\n\n            unchecked {\n
_mint(to[i], ++_soldTokens);                \n                if
(soulbound) _soulbound[_soldTokens] = true;\n                ++i;\n
}\n        }\n    }\n\n    /// @notice Only owner can call this function.
Free of charge. Mints sizeof(`to`) to `to` addresses with random
```

```
tokenIds. Requires `minting type` to be `random`.\n    /// @param to The
addresses of the NFTs receivers\n    /// @param soulbound True if the NFT
is a Soulbound Token (SBT). If set, it can't be transferred.\n
function airdropRandom(address[] memory to, bool soulbound)\n
external\n        payable\n        override\n        onlyOwner\n    {\n
uint256 toLength = to.length;\n        if (_mintingType !=
MintingType.RANDOM) revert InvalidMintingType();\n\n        if
((_soldTokens + toLength) > _collectionSize) revert
CollectionSoldOut();\n\n        unchecked {\n            _soldTokens +=
uint32(toLength);\n        }\n\n        bytes32 randomSeed =
blockhash(block.number - 1);\n\n        for (uint256 i; i < toLength; )
{\n            uint256 newTokenId = _randomTokenId(randomSeed, i);\n
_mint(to[i], newTokenId);\n            if (soulbound)
_soulbound[newTokenId] = true;\n            unchecked {\n
++i;\n            }\n        }\n    }\n\n    /// @notice Only owner can
call this function. Free of charge. Mints sizeof(`to`) to `to` addresses
with specified tokenIds. Requires `minting type` to be `specify`.\n
/// @param to The addresses of the NFTs receivers\n    /// @param
tokenIds An array of the specified tokens. They must not be minted,
otherwise, it will revert.\n    /// @param soulbound True if the NFT is a
Soulbound Token (SBT). If set, it can't be transferred.\n    function
airdropSpecify(\n        address[] memory to,\n        uint256[] memory
tokenIds,\n        bool soulbound\n    ) external payable override
onlyOwner {\n        uint256 toLength = to.length;\n        if
(_mintingType != MintingType.SPECIFY) revert InvalidMintingType();\n\n
if (_soldTokens + (tokenIds.length) > _collectionSize) revert
CollectionSoldOut();\n        if (toLength != tokenIds.length) revert
InvalidInputSizesDontMatch();\n\n        unchecked {\n
_soldTokens += uint32(toLength);\n        }\n\n        for (uint256 i; i
< toLength; ) {\n\n            if (tokenIds[i] == 0 || tokenIds[i] >
_collectionSize) revert InvalidTokenId();\n            _mint(to[i],
tokenIds[i]);\n            if (soulbound) _soulbound[tokenIds[i]] =
true;\n            unchecked {\n                ++i;\n            }\n
}\n    }\n\n    /// @notice Mints `amount` of NFTs to `to` address with
optional specified tokenIds. This function must be called only if a valid
`signature` is given during a whitelisting/presale.\n    /// @param to
The addresses of the NFTs receivers\n    /// @param tokenIds An optional
array of the specified tokens. They must not be minted, otherwise, it
will revert. Only used if minting type is `specify`.\n    /// @param
freeMinting True is minting is free\n    /// @param customFee Zero is fee
is different from `mintingPrice`.\n    /// @param maxAmount Max Amount to
be minted with the given `signature`.\n    /// @param amount Amount to
mint.\n    /// @param soulbound True if the NFT is a Soulbound Token
(SBT). If set, it can't be transferred.\n    /// @param signature Valid
`signature` for the presale/whitelist.\n    function mintPresale(\n
address to,\n        uint256[] memory tokenIds,\n        bool
freeMinting,\n        uint256 customFee,\n        uint256 maxAmount,\n
uint256 amount,\n        bool soulbound,\n        bytes calldata
signature\n    ) external payable override {\n        if (amount == 0)
revert InvalidAmount();\n\n        _usedAmountSignature[signature] +=
amount;\n        if (_usedAmountSignature[signature] > maxAmount) revert
NotEnoughAmountToMint();\n\n        if (_soldTokens + amount >
_collectionSize) revert CollectionSoldOut();\n\n        if (_currentPhase
== SalePhase.CLOSED) revert PresaleNotOpen();\n\n        address signer =
ECDSAUpgradeable.recover(\n
ECDSAUpgradeable.toEthSignedMessageHash(\n                keccak256(\n
abi.encodePacked(\n                        this.mintPresale.selector,
\n                         address(this),
\n                         block.chainid,
\n                         to,\n                        freeMinting,\n
customFee,\n                         maxAmount,\n
soulbound\n                    )\n                )\n            ),\n
signature\n        );\n\n        if (signer != N2M_PRESALE_SIGNER &&
signer != owner()) revert SignatureMismatch();\n\n        if
(freeMinting) {\n\n            if (msg.value != 0) revert
InvalidMintFeeForFreeMinting();\n        } else {\n\n            if
(customFee == 0) customFee = _mintPrice;\n
_requirePayment(customFee, amount);\n        }\n\n        if
(_mintingType == MintingType.SPECIFY) {\n\n            if
(tokenIds.length != amount) revert InvalidInputSizesDontMatch();\n
_mintSpecify(to, tokenIds, soulbound);\n        } else if (_mintingType
== MintingType.RANDOM) {\n            bytes32 seed =
keccak256(abi.encodePacked(signature));\n            _soldTokens +=
uint32(amount);\n            _mintRandom(to, amount, seed, soulbound);\n
} else if (_mintingType == MintingType.SEQUENTIAL) {\n
_mintSequential(to, amount, soulbound);\n        } else {\n\n
revert PresaleInvalidMintingType();\n        }\n    }\n\n    /// @notice
Returns the minting price of one NFT.\n    /// @return Mint price for one
NFT in native coin or ERC-20.\n    function mintPrice() external view
returns (uint256) {\n        return _mintPrice;\n    }\n\n    /// @notice
Returns the current total supply.\n    /// @return Current total
supply.\n    function totalSupply() external view returns (uint256) {\n
return _soldTokens;\n    }\n\n    /// @notice Max amount of NFTs to be
hold per address.\n    /// @return Max per address allowed.\n    function
maxPerAddress() external view override returns (uint16) {\n        return
_maxPerAddress;\n    }\n\n    /// @notice Returns how much royalty is
owed and to whom, based on a sale price that may be denominated in any
unit of exchange. The royalty amount is denominated and should be paid in
that same unit of exchange.\n    /// @param salePrice The sale price\n
/// @return receiver the receiver of the royalties.\n    /// @return
royaltyAmount the amount of the royalties for the given input.\n
function royaltyInfo(\n        uint256, \n        uint256 salePrice\n
) external view virtual returns (address receiver, uint256 royaltyAmount)
{\n\n        return (address(this), uint256((salePrice * _royaltyFee) /
100_00));\n    }\n\n    function _requirePaymentWithAffiliates(uint256
amount, address affiliate)\n        internal\n    {\n        uint16
currentUserDiscount;\n        uint16 currentAffiliatePercentage;\n
if (_affiliatesInfo[affiliate].enabled) {\n
currentUserDiscount = _affiliatesInfo[affiliate].userDiscount;\n
currentAffiliatePercentage =
_affiliatesInfo[affiliate].affiliatePercentage;\n        } else {\n
currentUserDiscount = _affiliatesInfo[address(0)].userDiscount;\n
currentAffiliatePercentage =
_affiliatesInfo[address(0)].affiliatePercentage;\n        }\n\n
uint256 discountMintPrice = ((100_00 - currentUserDiscount) * _mintPrice)
/ 100_00;\n        _requirePayment(discountMintPrice, amount);\n
if (affiliate != address(0)) {\n            uint256 affiliateAmount =
(currentAffiliatePercentage * discountMintPrice * amount) / 100_00;\n
_pendingTotalAffiliatesBalance += affiliateAmount;\n
_pendingAffiliateBalance[affiliate] += affiliateAmount;\n            emit
AffiliateSell(affiliate);\n        }\n    }\n\n    function _checkPhase()
private {\n\n        if (_currentPhase != SalePhase.PUBLIC) {\n
if (_currentPhase == SalePhase.DROP_DATE) {\n               if
(block.timestamp >= _dropDateTimestamp) {\n
_currentPhase = SalePhase.PUBLIC;\n
delete(_dropDateTimestamp);\n                } else {\n
revert WaitUntilDropDate();\n                }\n            } else if
(_currentPhase == SalePhase.DROP_AND_END_DATE) {\n                if
(block.timestamp < _dropDateTimestamp) {\n                    revert
WaitUntilDropDate();\n                }\n                if
(block.timestamp >= _endDateTimestamp) {\n                    revert
SaleFinished();\n                }\n            } else {\n\n
revert PublicSaleNotOpen();\n            }\n        }\n    }\n\n
function _requirePayment(uint256 p_mintPrice, uint256 amount) internal
{\n        if (_isERC20Payment == false) {\n\n            if (msg.value
!= (p_mintPrice * amount)) revert InvalidMintFee();\n        } else {\n\n
if (p_mintPrice == 0) return;\n            uint256 totalAmount =
p_mintPrice * amount;\n\n
SafeERC20Upgradeable.safeTransferFrom(\n
IERC20Upgradeable(_erc20PaymentAddress),\n                msg.sender,\n
address(this),\n                totalAmount\n            );\n        }\n
}\n\n    modifier onlyAllowedOperator() {\n\n
_isOperatorAllowed(msg.sender);\n        _;\n    }\n\n    modifier
onlyAllowedOperatorApproval(address operator) {\n
_isOperatorAllowed(operator);\n        _;\n    }    \n\n    function
_isOperatorAllowed(address operator) private {\n        if
(_operatorFilterStatus == OperatorFilterStatus.ENABLED_EXISTS) {\n\n
operatorFilterRegistry.isOperatorAllowed(address(this), operator);\n\n
} else if (_operatorFilterStatus ==
OperatorFilterStatus.ENABLED_NOT_INITIALIZED) {\n\n            if
(address(operatorFilterRegistry).code.length > 0) {\n                try
operatorFilterRegistry.registerAndSubscribe(address(this),
DEFAULT_SUBSCRIPTION) {\n\n                    _operatorFilterStatus =
OperatorFilterStatus.ENABLED_EXISTS;\n
operatorFilterRegistry.isOperatorAllowed(address(this), operator);\n
} catch {\n                }\n            }\n        } \n\n    }\n\n
fallback() external payable\n    {\n        address libraryAddress =
LIBRARY_ADDRESS;\n\n        assembly {\n\n            calldatacopy(0, 0,
calldatasize())\n\n            let result := delegatecall(\n
gas(),\n                libraryAddress,\n                0,\n
calldatasize(),\n                0,\n                0\n            )\n\n
returndatacopy(0, 0, returndatasize())\n\n            switch result\n\n
case 0 {\n                revert(0, returndatasize())\n            }\n
default {\n                return(0, returndatasize())\n            }\n
}\n    }\n\n    receive() external payable {}    \n}\n"
    },
    "contracts/N2MVersion.sol": {
      "content": "/** ---------------------------------------------------
------------------------- //\n *
//\n *                                        .:::.
//\n *                                     .:::::::.
//\n *                                     ::::::::.
```

```
//\n *                                 .:::::::::.
//\n *                             ..:::.               ..
//\n *                           .::::.                ::::..
//\n *                      ..:::..                    ::::::::.
//\n *                   .::::.                        :::.  ..:::.
//\n *               ..:::..                           :::.      .:::.
//\n *            .::::.                               :::.
.:::..           //\n *         .:::..               ..
:::.            .::::.        //\n *     .::::.               ..:::=-
::::               ..:::.     //\n *    :::.               .::::::::===:
:::::.                .::::   //\n *   .::.            .::::::::::::=====.
::::::::::.             .::.  //\n *   .::
.::::::::::::::::=======.         ::::::::::::::..         ::.  //\n *
.::        .::::::::::::::::=========-        :::::::::::::::::
::.  //\n *   .::        .::::::::::::::::===========:
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::============:     :::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::==============.   :::::::::::::::::
::.  //\n *   .::        .::::::::::::::::===============-.
:::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::=================:::::::::::::::::         ::.  //\n *
.::        .::::::::::::::::==================-:::::::::::::::
::.  //\n *   .::        .::::::::::::::::==================-
::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::==================-::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::=================-:::::::::::::::
::.  //\n *   .::        .:::::::::::::::: .-================-
::::::::::::::::        ::.  //\n *   .::        .::::::::::::::::
.==============-::::::::::::::::        ::.  //\n *   .::
.::::::::::::::::     :============-::::::::::::::::        ::.  //\n *
.::        .::::::::::::::::       :==========-::::::::::::::::
::.  //\n *   .::        .::::::::::::::::        .-========-
::::::::::::::::        ::.  //\n *   .::          .::::::::::::::
.=======-:::::::::::::.          ::.  //\n *   .::.
.::::::::::            .=====-::::::::::..            ::.  //\n *
:::..              ..::::::              :===-::::::..              .:::.
//\n *      .:::..               .:::                -=-:::.
.::::.    //\n *         .::::.            .:::                 ..
.::::.       //\n *            .::::.         .:::
..:::.           //\n *                .:::.      .:::
.::::.              //\n *                   .:::..  .:::
..:::..                 //\n *                      .::::.:::
.::::.                     //\n *                         ..::::
..:::..                        //\n *                              .:
.::::.                            //\n *
:::::.::::.                               //\n *
::::::::.                                  //\n *
:::::::.                                   //\n *
.::::.                                    //\n *
//\n *
//\n *   Smart contract generated by https://nfts2me.com
//\n *
//\n *   NFTs2Me. Make an NFT Collection.
//\n *   With ZERO Coding Skills.
```

```
//\n *
//\n *   NFTs2Me is not associated or affiliated with this project.
//\n *   NFTs2Me is not liable for any bugs or issues associated with
this contract. //\n *   NFTs2Me Terms of Service:
https://nfts2me.com/terms-of-service/            //\n * ----------------
--------------------------------------------------------------- */\n\n///
SPDX-License-Identifier: UNLICENSED\npragma solidity ^0.8.18;\n\nimport
\"@nfts2me/contracts/important/README.sol\";\n\n/// @title NFTs2Me.com
Smart Contracts Version\n/// @author The NFTs2Me Team\n/// @notice Read
our terms of service\n/// @custom:security-contact
security@nfts2me.com\n/// @custom:terms-of-service
https://nfts2me.com/terms-of-service/\n/// @custom:website
https://nfts2me.com/\nabstract contract N2MVersion is Readme {\n    ///
@notice Current version of the nfts2me.com contracts.\n    function
n2mVersion() virtual external pure returns (uint256) {\n        return
1000;\n    }\n\n}\n"
    }
  },
  "language": "Solidity"
}
```